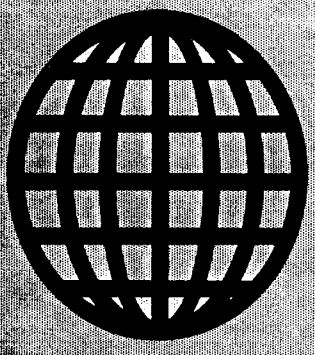
Providing Support Around The World



The Computer Journal

Issue Number 75 September/October 1995 US$4.00

Embedded Control Using the Standard BUS

Small System Support

T9600 Source Code

European Beat

Real Computing

High-Speed Serial I/O - PCPI Applicard

Dr. S-100

Eprom Simulator

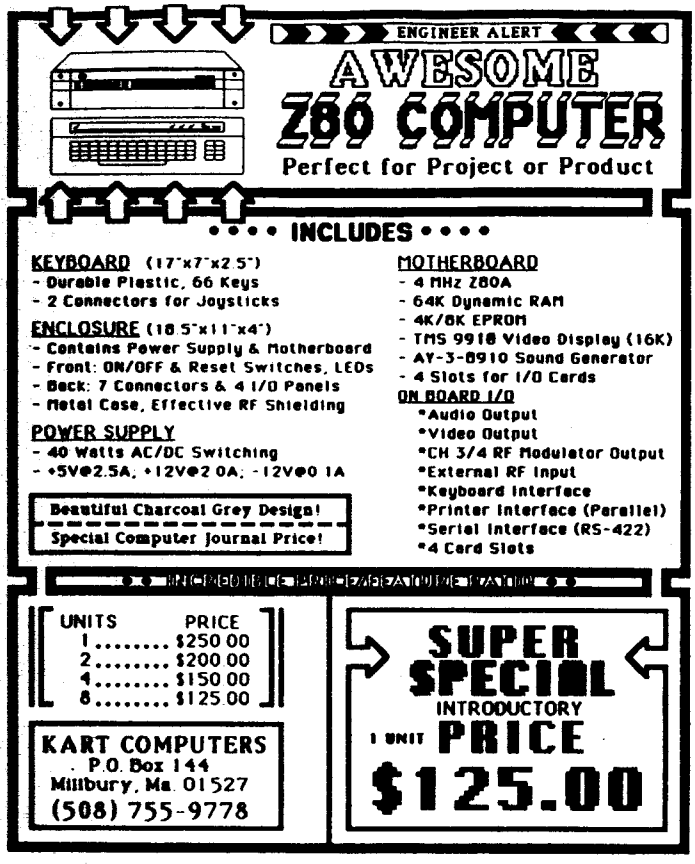
Disk I/O in Forth, F83

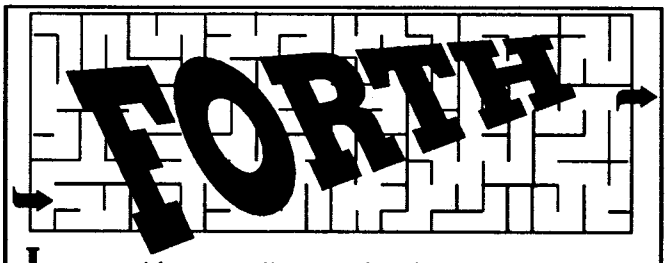
Centerfold - STD BUS I/O

The Computer Corner

ISSN # 0748-9331

TCJ - For Having Fun With Old Computers!





Journey with us to discover the shortest path between programming problems and efficient solutions.

The Forth programming language is a model of simplicity: In about 16K, it can offer a complete development system in terms of compiler, editor, and assembler, as well as an interpretive mode to enhance debugging, profiling, and tracing.

As an "open" language, Forth lets you build new control-flow structures, and other compiler-oriented extensions that closed languages do not.

Forth Dimensions is the magazine to help you along this journey. It is one of the benefits you receive as a member of the non-profit Forth Interest Group (FIG). Local chapters, the GEnie™ Forth RoundTable, and annual FORML conferences are also supported by FIG. To receive a mail-order catalog of Forth literature and disks, call 510-89-FORTH or write to:
Forth Interest Group, P.O. Box 2154, Oakland, CA 94621.
Membership dues begin at $40 for the U.S.A. and Canada.
Student rates begin at $18 (with valid student I.D.).

GEnie is a trademark of General Electric.

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low as $100.00
Developer Packages
as low as $200.00(a $50.00 Savings)

A New Project
Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files.

Get It To Market--FAST
Don't wait until the hardware is finished to debug your software. Our Simulators can test your program logic before the hardware is built.

No Source!
A minor glitch has shown up in the firmware, and you can't find the original source program. Our line of disassemblers can help you re-create the original assembly language source.

Set To Go
Buy our developer package and the next time your boss says "Get to work.", you'll be ready for anything.

Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

BROAD RANGE OF SUPPORT

- Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	Intel 80C196

- All products require an IBM PC or compatible.

So What Are You Waiting For? Call us:

PseudoCorp

Professional Development Products Group

921 Country Club Road, Suite 200

Eugene, OR 97401

(503) 683-9173 FAX: (503) 683-9186 BBS: (503) 683-9076

SAGE MICROSYSTEMS EAST

Selling and Supporting the Best in 8-Bit Software

Z3PLUS or NZCOM (now only $20 each)
ZSDOS/ZDDOS date stamping BDOS ($30)
ZCPR34 source code ($15)
BackGrounder-ii ($20)
ZMATE text editor ($20)
BDS C for Z-system (only $30)
DSD: Dynamic Screen Debugger ($50)
4DOS "zsystem" for MSDOS ($65)
ZMAC macro-assembler ($45 with printed manual)

Kaypro DSDD and MSDOS 360K FORMATS ONLY
Order by phone, mail, or modem and use Check, VISA, or MasterCard. Please include $3.00 Shipping and Handling for each order.

Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469
(617) 965-3552 (voice 7PM to 11PM)
(617) 965-7259 BBS

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler

Technical Consultant
Chris McEwen

Contributing Editors
Herb Johnson
Charles Stafford
Brad Rodriguez
Ronald W. Anderson
Helmut Jungkunz
Ron Mitchell
Dave Baldwin
Frank Sergeant
JW Weaver
Richard Rodman
Jay Sage
Tilmann Reh

The Computer Journal is published six times a year and mailed from *The Computer Journal*, P. O. Box 535, Lincoln, CA 95648, (916) 645-1670.

Opinions expressed in *The Computer Journal* are those of the respective authors and do not necessarily reflect those of the editorial staff or publisher.

Entire contents copyright © 1995 by *The Computer Journal* and respective authors. All rights reserved. Reproduction in any form prohibited without express written permission of the publisher.

Subscription rates within the US: $24 one year (6 issues), $44 two years (12 issues). Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 535, Lincoln,CA 95648.

Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used trademarks are acknowledged, and we apologize for any we have overlooked.

Apple II, II+, IIc, IIe, Lisa, Macintosh, ProDos; Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. DateStamper, BackGrounder ii, Dos Disk; Plu*Perfect Systems. Clipper, Nantucket; Nantucket, Inc. dBase, dBASE II, dBASE III, dBASE III Plus, dBASE IV; Ashton-Tate, Inc. MBASIC, MS-DOS, Windows, Word; MicroSoft. WordStar; MicroPro International. IBM-PC, XT, and AT, PC-DOS; IBM Corporation. Z80, Z280; Zilog Corporation. Turbo Pascal, Turbo C, Paradox; Borland International. HD64180; Hitachi America, Ltd. SB180; Micromint, Inc.

Where these and other terms are used in *The Computer Journal*, they are acknowledged to be the property of the respective companies even if not specifically acknowledged in each occurrence.

TCJ The Computer Journal

Issue Number 75 September/October 1995

Editor's Comments 2

Reader to Reader 3

Dr. S-100 8
The Mailbag.
By Herb Johnson.

The European Beat 11
East German Z80.
By Helmut Jungkunz.

Real Computing 14
Rick moved and more.
By Rick Rodman.

Small System Support 16
C and assembly language tutorial.
By Ronald W. Anderson.

Embedded Control Using the STD BUS 21
Details of the Standard BUS.
By Bill Kibler.

Center Fold 25
STD BUS I/O.

EPROM Simulator 29
Hardware help using ROMs.
By Terry Hazen.

Disk I/O in Forth 33
Part 2 about F83 Forth Disk I/O.
By Walter Rottenkolber.

High-Speed Serial I/O for the PCPI Applicard 38
Apple][support.
By John D. Baker.

T9600 Source Code 43
Last part of Small Tools support.
By Calvin McCarthy.

Support Groups for the Classics 46

The Computer Corner 50
By Bill Kibler.

EDITOR'S COMMENTS

Welcome to issue 75 and my next to last as current editor. With that starting line, I better explain before I tell you what is in this issue.

Time Off

I have decided I need time off to spend with my family and hopefully make some money to replace what I lost doing *TCJ*. Now *TCJ* breaks even with respect to inflow versus outflow. I however took one day a week off without pay in hopes of making it back through the journal. That has never even come close to happening. Partly because I never seem to have time to make it happen, and also because "collecting" is still not widely thought of as an aspect of computing.

One problem with *TCJ* is not being able to afford to be on the newsstand. That would cost about $4000 each issue, mostly as pure advertising expense. We have never made that kind of extra money, and I certainly don't have an extra $30,000 to spend each year. That means we (as a magazine) will be small for some time.

I had been considering this for some time and finding a replacement for myself was the major stumbling block. I found David Baldwin many years ago at one of the computer meetings and we have been friends ever since. Some time ago I asked him about taking *TCJ* and he was not very interested at the time. The idea however grew on him and he decided to ask more questions. About that time I decided to say no more after #76. We talked, I showed him things, he pondered more. When I came back from a few weeks off, he was hooked and has been turning and burning ever since.

Dave has been a Z80 embedded controller builder for many years and has his own BBS. He is active on the internet, has plenty of ideas, and importantly his kids are grown and almost gone from home (I'm a late bloomer - mine is only 6). He is usually in his office every morning and often all day working on client projects. He has already taken *TCJ* material and is busy working on getting ready to take over. You can see some of his work by checking out our Web pages on www.psyber.com/~tcj.

STD BUS SPECIAL

Now this issue has some special work on the STD BUS. I explain what it is and give you the centerfold on a simple I/O example. Our alternating focus makes this one on embedded controls and as such I have tried making most articles center on that.

Our letters to the editor has some mail lost from other issues and a few new items as well. I am trying to catch up before handing my mail over to David, so plenty of letters here and in next issue.

Next up is Helmut's Europe report. He has been talking about East German machines and even includes some pictures. I have more pictures to show, and will try and get them in later as he continues to cover the European Beat.

Dr S-100 is next with more mail bag, and I talked to Jade Computer who stated their S-100 products are basically public Domain now. That mean personal not commercial use allowed. This means Herb can now do the Jade Bus article everyone has been asking for.

Ron Anderson gives us more C and Assembly help and a few tid bits as well. Rick Rodman just moved and sounds like he is setting up a great computer lab.

The Standard Bus feature is next up and runs into the centerfold. For STD use or Z80 use is the EP-SIM or Terry Hazen's article on building your own EPROM simulator. He used it on his YASBEC, but you can use it on most any machine. Also read his notes on using the public domain version of PADs.

Part two of Disk I/O in Forth is next with the F83 definitions and source code. Next for you Apple folks, John D. Baker gives us high speed serial using the PCPI Applicard. And I follow this with the last installment of Clavin McCarthy's Small Tools, his source code for the T9600 terminal interface.

Lastly is little old me and my Computer Corner. This time I explain a little more about the changes and show you the photo of me giving David Jaffe his 1994 Computing Hero award. By the way, know of some one for the next award?

The WEB

For those of you who haven't seen our WEB PAGE, start tooling up to do so. Dave and I have been hard at it, making changes almost daily. The best place to find out about back issues is now there. Dave is also keeping all the same files on his BBS and has uploaded some to GENIE and CompuServe.

Dave is getting pretty active on the UseNets and trying to get the word around. Since the page has been active, over 3000 references have been made to it. We are getting a lot of lookers and our references and links to other locations are very popular. If you know of places of interest to our readers send us an e-mail so we can put the information on line. Try:
e-mail tcj@psyber.com
dibald@netcom.com
http://www.psyber.com/~tcj

Sounds like fun to me, and thanks for staying with *TCJ*! Bill.

READER to READER

Letters to the Editor
All Readers
MINI Articles

Dear Sir:

I would like to become a subscriber to your excellent magazine. I would also like to receive a back-issue copy of TCJ#65 which contains the ZX80/81 centerfold. Enclosed please find a postal money order for the total of $29.25 ($24 subscription + $4 back-issue +1.25 s&h). I have been looking for a long time for a publication such as yours which includes systems other than IBMs and Mac's. I got my start with a Timex/ Sinclair 1000 (I still own two - in storage).

I moved on to TRS-80 III's & IVs which was my first experience with CP/M. While in college I worked as a programmer on a project that linked an IBM-XT class to a custom-made Z80-based single-board-computer which controlled an engraving system. I miss the Z80!

Anyway, I look forward to reading more of your publication. I have a few ideas/ suggestions/requests that I will e-mail to you when I get them organized. I have also recently obtained my first microcontroller (an Xplor-32 made by Blue-Earth Research) and I am planning on giving it sound capability with a AY-3-8910 programmable sound generator. If I really do manage to avoid my procrastinating nature and finish this project, I will submit an article. That's all for now.

Sincerely, Arthur C. Jones
(ajones@omni.voicenet.com)

Well have no fear we are here to keep Z80 alive and well. I am not sure, but I think the Xplor-32 is not a Z80 machine, but since you have indicated you did cut your teeth or get your experience on Z80, I guess trying something else out is ok, but only as long as you send us a review article.

Thanks for the renewal and by the way, there are still lots of machines that get instructions from a PC, but are Z80's that are really doing all the work. I think the Z80 is still number one processor sold, since almost all telephone switches use a single Z80 for each incoming phone line. Lets see that is how many phone is use... BDK.

Sir:
Congratulations on a fascinating magazine. I received Issue #73 and found it very interesting. Of course I was particularly interested in seeing how my Small Tools article looked in print. Issue #72 published the Pygmy tools part of the article but didn't include the code listing. I was hoping that the tools specific code would accompany the text because I am sure that the article would have been more readable with the code. If I did not include the code in a text file then it is my mistake.

On another note, I would like to see the package including Pygmy, Pygtools, the article,, and the code distributed on the BBS. I am wondering what you have posted and what I would be able to post considering your copyright. I would appreciate it if you would give me permission to post the whole package if you have not done this.

Thank you for your work to create this unique forum for information-nation and look forward to the future issues of the magazine.

Calvin McCarthy
E-Mail: ah607@freenet.carleton.ca

Clavin, thanks for the wake-up letter and your article. I think I have failed again and not uploaded it yet. I usually put them on the DIBS BBS and try Genie as well. I will be putting them on my FTP space on psyber.com and have the web page link to them. All that takes time and also why your other parts have not been printed. It is in this issue. I am trying not to overload a single issue these days.

As to rights, you still retain them, and can freely distribute them as you want. Since your reward is just a few minutes or pages of fame, no money, I find it hard to feel good about retaining any control over your hard work. I hope too that anyone with a real interest in what you have done, will send you a few bucks for your time, of course better yet might be a few contract programming jobs?

Take care and thanks. Bill.

Dear Bill:

Evidently the Motorola 68HCll microprocessor is fairly popular, and I have a tip that might help someone using it at low frequency.

I tried running an 8 Mhz development kit at 32.768 Khz and ran into a signal isolation problem that prevented the clock divider from working. Low frequency crystals cannot take much power, which is why the drive must go through a resistor. Also, the load capacitors will present a high impedance and any stray capacitive coupling is very bad news. It seems that the crystal amplifier input pin and the buss "E" clock pins are close together, and I think my board coupled enough signal that the clock divider on the chip would turn itself off when it

started to switch. An oscilloscope showed a glitch at the crystal amplifier output (XTAL pin) that may partly have been due to coupling capacitance between the EXTAL and E pin in the LCC socket. Whatever the reason, the only fix I found was hanging a 100 Pf capacitor between XTAL and ground. This causes a small increase in power drain which I can live with. Using a 3 volt battery supply (alkaline "D" cells) and single chip mode the drain is about 170 Microamps. According to Duracell the batteries should give 17 ampere hours, but for reasonable output I am assuming 8. At this capacity the board should run at least 5 years.

At 5 volts the crystal drive (XTAL pin) is close enough to sinusoidal that the CMOS amplifier is on a significant amount and drains a few milliamperes. Reducing the supply to 3 volts makes the drive more nearly square and the stage takes a lot less current. Since the HC11 is built with CMOS you cannot leave inputs floating or the supply current goes way up anyway.

I use the boot mode loader to start up the board, and at the default baud rate (32) it takes a few minutes to get started. Setting the rate to 512 (highest available) helps, as does transmitting Motorola "S" records in binary format.

Your truly, Frank Wilson

Boy you can say the 68xx chips are popular, try and buy some. The distributors aren't even quoting prices. Your project sounds very interesting, how about some more facts, in an article? Thanks Frank and keep up the hacking! Bill.

Hello again! Thanks for the LAST ISSUE ** sticker... A year went by much too fast, but I am making progress toward the college degree. And then TCJ; the Journal truly is one-of-a-kind. Sometimes I am looking for things in it that seem to be missing, things that I like to do in my free time, as well as perhaps smaller applications geared toward a professional career. I enjoy F. Sergeant's PC-XT Corner, particularly the explanations of battery-backed RAM. And the ongoing series by B. Rodriguez entitled MOVING FORTH has been most enlightening.

I must notes that after buying EVERY back issue (which I will never regret), it is somewhat saddening to see what was the mainstream of 'hacking' just a few years ago, which is just about where I am today, and how so many advertisers have disappeared. Things appear to be changing, but I speculate that this is not in fact the case.

The modern world of color monitors and OPC Hatrodsm, as I call them, is often all too attractive to an otherwise-dedicated computer hardware hobbyist. It serves to allure, suck up their money, and zap precious time. I don't have a statistic, but I could safely assume that far fewer of us today are as 'hardwareliterate' as the (kids) who were there in the mid 70's, in the heart of the personal computer revolution, actually building them from scratch and writing their own cassette driver software.

So it might not be a change in method - it certainly isn't a change in technology - which keeps the hardware articles from flowing. I think it might be simple distractions perhaps coupled with disorientation from watching too much color television.

Enough... Here is a bank check for $24.00; thanks again for TCJ. Cordially, Douglas P. Beattie, Jr.

Thanks Douglas for those comments and here is some news for you. Things will change some with David taking over, but he has more time to make sure we stay old-fashioned. He will be looking forward to your comments.

If I remember right you wanted to do stuff with the Rockwell F65F11, well I got this e-mail the other day:

Hi!

I was browsing and I noticed a comment about last issue centerfold being a Rockwell F65F11 board. I made Rockwell's "Low Cost Development Board" for them for years. I sold several hundred of them when I lived/worked in Huntington Beach. They gave me their artwork and tooling and sent all their customers to me. Turned out to be a really great deal for me. It also was what got me into Forth and I have always been grateful for that.

I've been a real fan of Forth ever since. Was the centerfold of that board or someone elses? Is it possible to get a copy of that issue? I've still got some of the Forth kernel roms and development roms left from those days as well as a lot of pc boards (blank and stuffed) for the 65F12 and the emulator board. Old Forthers never die!

Regards, Art Horne ahorne@aa.net

I said:

Well great that you have boards, have one reader looking frantically for one. Will give him your email address and would love to have it so can publish a follow up on the boards. Rockwell is lost when the subject is asked, they forget quickly what they did years in the past....

Give me your address and I will send you a sample of TCJ, the one with the 68F11 in it. We send samples out, then hopefully you will subscribe.

Hi Bill,

My address is Horne Electronics, Inc. 20924 NE 165th. St., Woodinville, WA 98072. Phone number is (206) 788-9718, FAX # (206) 788-9833. Feel free to publish that any way you want.

I also have a few of the old RSC FORTH manuals. I'm not sure if I have any of the main boards but I know I have some of the adapter cards and some roms, etc. I may have the original artwork films for the main board but I haven't seen them in quite a while.

Joe Hance and Randy Dumse were the originators of that Forth for Rockwell. I don't know where Joe is now but Randy is "Mr. New Micros" in Dallas. I used to live a few miles from Rockwell when they came out with that chip. I sold quite a few of those boards both as kits and as assembled units. I still love Forth and try

not to program in anything else. I'll keep an eye out on your WWW page. Thanks for the reply. Best Regards, Art Horne (K6KFH), May the FORTH be with you!

Well that sure is some information and thanks Art, I think you will probably get a few calls out of this. So New Micros Forth is similar to F65F11...hummm.

From: GARY RATLIFF
Subj: Assistant Editor App

Here is a check in the amount $40.00. This is for the most recent ten back issues. Per an earlier message you mentioned that you would be glad to pick up the shipping charges.

This should give me a handle on what has been happening. Then with a sample of number 74 I would be up to speed.

I have already been scouting the old computer haunts for perhaps picking up one of these TRS 80 with level 1 Basic. This is from 1978 as I took a course at Missippi College on Microcomputing and the level one BASIC is what was taught. It was here that I also went to The Oldee Computer Shoppe in Metarie Louisiana and purchased my very first computer an 8k Commodore PET. (The keyboard sucked but the BASIC was a much more powerful version than the few commands offered in Level 1.) In my old books I still have the manual for level one Basic which I used during that course. Also have talked to another computer junkie who is apt to have one of these.

Then too as RS is still in business I might be able to write Tandy and get some schematics and other materials from the horses mouth. The back issues should give me a good idea of how the antique feature is being done.

Dave recently sent me a packet of all the back issues from The Z-Letter. While I was moderating I spent so much time reading the various cpm echos that I let reading any magazines slide.

gary

Well great Gary and Welcome as our FIRST support editor of classic systems. Issue number 77 will be the one you get to show your stuff in, and it sounds like you have started researching the Radio Shack line. Don't forget the CoCo as they are very popular still, more so I think that the Z80 machines. My feeling is there were many better Z80 machines than RS made, but not many other 6809 machines that were more powerful for the bucks.

Thanks for helping and Dave Baldwin will certainly be talking with you soon. BDK.

Dear Bill Kibler,,

Enclosed find my renewal check for the next two years. I know you've been struggling with the magazine, and I wanted to send more than a check to let you know how much I value the hard work you've put into it. I've kept and maintained all the machines I've written with over the years and as a result have what has become, I guess, a collection. NorthStar Horizon 8/16s running TurboDOS, NorthStar Advantage 8/16s running CP/M and DOS 2.0,, Epson QX-10s running CP/M,, Valdocs & DOS 3.3, and a NEC PC-8500 CP/M laptop with all the add-on bells and whistles. Perhaps in the future I might write something about some of them. I'm a screenwriter working in the movie and television business, and although my workaday machine is now a Pentium-90-driven monster, I still have a little comer of its 2 gigabyte drive for Simeon Cran's simulator and some of my old software favorites, running a good deal faster than they ever did on any of my many Z-80s. Best of luck to you.

Sincerely,,
JHR, Beverly Hills, CA 90213-1823

Would love to have something from you about how you use and enjoy the MYZ80 program. It seems very popular, but we haven't had a "how I do it" report yet. Interested?

In #74 TCJ classified Mr. David Kruszyna of Eastpointe, MI wanted a hard drive and modem for an Apple IIGS. There are still several retailers serving the Apple][s but arguably the largest is in his state - Quality Compters, 20200 Nine Mile Road, St. Claire Shores, MI 48080 (1-800-890-8263).

Their latest catalog (APV3) lists a 2400 Baud Hayes Compatible Modem for $74.95. With software and a cable they want $139.95. In regards the hard drive, I believe most use a SCSI interface. I believe any Macintosh SCSI hard drive will work after reformatting. All that's needed are the SCSI utilities from the GS system disks and a SCSI card. Quality is selling a preformatted 270MB Quantum with SCSI card, cables, manual and system software installed for $349.95. They also sell a 4MB memory expansion board for $199.95.

Anyone wanting some technical info, books or programming products check out Byte Works, Inc. 8000 Wagon Mound Drive N.W., Albuquerque, NM 87120 (505-898-8183) for reasonable prices. Great products! Finally, seemingly the last Apple][magazine is GS+ Magazine, P.O. Box 15366, Chattanooga, TN 37415-0366 (615-332-2087). As the name implies, it is almost all GS oriented.

I hope this helps anyone with the greatest little Apple][ever built!

Steve Fuesting

Thanks Steve. We do get plenty of Apple requests and are looking for someone to talk about them on a regular basis. Might you be interested? Our needs are simple, answer reader questions and provide background facts and hardware details. There are still plenty of Apples running and needing help! Thanks. Bill.

Dear Bill;

I have some words of praise for the Amstrad PCWB256. Recently I was corresponding with Emmanuel Roche of France. We both use the Epson QX-10 computer. His Epson uses the European disk format, mine uses the American disk format.

My reply to his last letter would be better

put on disk, rather than several sheets of hard copy. This way I could submit my version of the software ready for him to run on his Epson QX-10. All that I needed was the European formatted floppy disk, which I both formatted and loaded on the Amstrad PCW.

My Amstrad PCW8256 has a 3" and 3 1/2" drive A, and a 5 1/4" drive B. Thanks to Bill Roch of Elliam Associates, who made the modifications.

Fortunately I have a program for the PCW which is called MFU (Multi Format Utility). I turn on the PCW, invoke MFU, which is menu driven. The first menu shows 12 single letter options. Type <L> to bring up the library manager menu. There I see "20 - Epson QX10 DS/DD". Next is "I)nstall a format from library". Type <I> and answer the question with <20>. Now at the bottom of the screen, I see a message which reads 'Drive B: is currently set to Epson QX10 DS/DD format.'. Drive "B-" will now emulate this format as marked. Incidentally, this one is the European format.

Next press the <EXIT> key to return to the first menu . Type <P> to produce a Stand-Alone format setup program, that will do what was described in the previous paragraph without loading MFU. In this case I answered the question with <EPSONEU>. EPSONEU.COM is now a Stand-Alone 2k file that will run on any PCW floppy disk. There is no longer any need to load that large (40k) MFU program, plus the required overlays, whenever I want to set the Amstrad PCW8256 drive B to emulate the European Epson QX10 DS/DD drive. Now typing <EPSONEU> sets drive B.

I used the "A)nalyse unknown disk in drive B" option, in the first menu of MFU, to tell me the format parameters of the American Epson QX-10 floppy disk. Next, I used the "E)dit format parameters for drive B" option, to create the EPSONAM4 format, by inserting the parameters just found. EPSONAM4 is the American Epson format with a skew factor of 4. It has been added to the MFU library of my Amstrad PCW8256. Now on the PCW I can format read or write either, the European, or the American, Epson QX-10 floppy disk. — AMSTRAD PCW8256 TO THE RESCUE! —

Sincerely yours, Robert L. Edgecombe, Atascadero, CA

Thanks Roger, I am sure looking forward to spending some time with my Amstrad, seems it just sits in storage since being editor eats up too much of my time. But I like all the utilities and features it has, especially the one you just mentioned (changed drive format by running a program you can create - neat!) and the few times I fired it up, it sure seemed like it would be fun to use. Oh well only one more issue to do, then vacation. Bill Kibler.

Dear Bill:

First I'll ask you to excuse my grammar. I never attended school on this side of the "Pond".

If I remember right I have been getting the *TCJ* for about a couple years now and I for sure don't hope you give up, if so *TCJ* will be the fifth magazine I'm losing because I'm mainly interested in Yesterdays computers. I have got three TS-1000, one TS clone PC 8300 (Lambda) one TS-2068 with printer, one Commodore 64 with Star NX-1000C I'm producing this letter on.

Besides those I have got one and 3/4 of a second one IBM XT clone. I have got the heck of a time to find card and other stuff that will run those. My last delight (headache) is a Atari PC 4 286, which I bought in a bankrupt sale, and I would be interested in to hear from anybody there can help with any kind of information, schematic, setup disk, and printed material.

I swallow anything about XT's in *TCJ*, especially #73 was super and I'm looking forward to some more.

Sincerely, A.J. Kammersgaard, Ottawa.

Well A.J. thanks for the letter and number 76 will be on XT and advanced systems. I may do some talking about cases and such, but had intended to talk more about alternatives to the XT, such as the PT68K-4 running SKDOS or OS-9. I'll see if I can't slide a little XT information in with the 68000 stuff.

Glad to see your collecting and don't worry, TCJ will be here for many more years, just not under my full time control. David Baldwin has lots of ideas to help out you XT people and classic users too. BDK.

Dear Mr. Kibler:

I'd like to start with a question: Where have all the small IDE hard drives gone? I'm referring to IDE hard drives 60MB and less (especially 20MB units). Outside of those that met with catastrophe while in service, I feel that there must be many, many of these small hard disks whose only failing is that they can't contain the bloated software foisted on the general public.

Given the extremely low prices of enormous hard disks, surely users have replaced their small-yet-still-functional drives. But what's become of the small ones that were removed? Now that we know how to interface IDE hard disks to our favorite small machines, these small-capacity drives would seem to be the ideal things to use. Alas, I haven't seen any used or refurbished small IDE drives anywhere. Granted, there's probably no visible market for them, but I hate to think that they've been melted down or worse, sent to a landfill!

If I could afford it, I'd buy up any and all such drives whenever I found some available just to make sure they went to good homes, but alas, I can't.

Related to this: While assembling a new case and power supply for the Conner CP-3022 (20MB) that I had been using with my Epson QX-10 via a host adapter based on Wayne Sung's control ROM, A little carelessness with the power supply connections caused me to destroy the hard disk. I figured that was the end of my IDE experiments, but fortunately, a fellow member of the CP/M-Houston Users' Group had a Conner CP-342 lying about that I could have. Although

I'm wasting over half the capacity, I at least get to continue working with 8-bit IDE on the QX-10.

This has also made me start thinking about a "HardCard" type disk for the QX-10. The limited space inside the QX-10 makes it difficult to put a 3.5" IDE hard disk inside, but one of the small-capacity 2.5" IDE hard disks would work just right (as long as said card is installed in Option Slot 5 in the QX-10. The smallest-capacity IDE I've seen available to me is a 60MB Conner. At $50, it's starting to look like I could dare to buy it... Of course finding the 2mm connectors is another matter....

Back this past spring, the same member of the user group gave me something of a treasure. It's a NorthStar Horizon, modified by TEI for Cray Research. It's the boot/diagnostic server to a Cray X-MP super computer. Rather than a conventional N* enclosure and motherboard, this beastie was mounted in a black 19-inch rack-mount case by Integrand using an ordinary 10-slot S100 passive backplane.

It had a fairly conventional suite of N* boards (ZPB Z-80A Processor Board, HRAM 64K, MDS Floppy controller, HRZ HD-5 Hard disk controller) And a Vector Graphic, Inc. Bitstreamer II [I/O2] board that had been modified by TEI so that the serial ports were addressed exactly like those of the N* motherboard. (Actually, after poking around with an ohm-meter, The Bitstreamer II serial ports were already addressed like the N* motherboard—TEI just rearranged a handshaking signal on one port and readdressed one of the parallel-port addresses to 0,1.)

Further, there were some Cray Research boards: CHNL BFR (Channel Buffer with 32K of SRAM), CRAY CTL (Channel Control Board), ER CHNL BRD (Error Channel Board). I received no disks with the machine at the time I took it home, but I got plenty of them at the next meeting, including all the original CP/M and N* HDOS disks (no manuals, though.)

The machine works great, including the 15MB hard disk. It booted right up the first time. I learned that the 32K on the Cray Channel Buffer board is required for hard-disk support under CP/M. It seems that N* HDOS runs in that 32K and manages the CP/M hard-disk filesystem as a file or files under HDOS. Pretty clever! Running it under CP/M is a snap, but all the hard disk utilities must be run under HDOS and beyond the information provided in a "cookbook" file on hard disk formatting and partitioning, I'm lost. I'm going to try to see if the member who gave me the N* has the docs for it.

Over the break between the spring and summer semesters, I designed and built a serial I/O expansion board for the PCPI AppliCard. Over an extended Independence-Day weekend I did the most debugging, testing, and programming of the board. I've submitted an article and schematic describing it.

During the week and a half break between the summer and fall semesters I said I was going to write that article on the board I mentioned. Instead, I became absorbed with mounting my Davidge DSB 4/6 (Actually a Davidge DSB 4000 Rev. B as I've since learned) single-board computer in an old Leading Edge PCLone case I was given. I now have a very neat and tidy setup with all of the Davidge's 4 serial ports, parallel printer port, 5.25/3.5" floppy and 8" floppy ports fully expanded and accessible.

The resulting machine has 2 80-track 5.25" drives (A: and B:, naturally) mounted in the new case and I have a pair of Shugart SA-860 " floppy drives plugged into the 8" floppy port as C: and D:. I also have a 720K 3.5" drive I can attach to an external floppy port as long as I unplug the 8" drives (due to address conflicts). It's nice and quiet compared to the abominable home-brew case and power supply it used to have. I had no idea it has power-on RESET until I put it in this new case! The only port left unexpanded is the Davidge's high-speed parallel port. I hope to remedy this as soon as I can build a suitable hard disk host adapter (the intended function of the port).

This past weekend (31 August-3 September) was probably the most interesting. With my Amiga, CDTV, and Davidge and my brother's 486 box all up at school, I had the space to set the NorthStar up and play with it some more. TEI's User BIOS segment implements Port C of the I/O2 board as PTP:/PTR:. So I experimented with transferring things through RDR: and PUN: with PIP for the first time! I'd read and heard about doing that for the longest time, but the N* is the first machine I've had that was actually capable of doing that.

With my new-found abilities, I managed to install Donald C. Kirkpatrick's ZCPR-D&J ZCPR1 module on the NorthStar. First, I assembled the test version that is relocated and run at 8000h. Using the PEEK command, I determined that BDOS in this 64K N* hard disk system resided at C500h. (I used my Apple //e CardZ180 system as the development platform then transferred the .HEX output to the N* using PIP on the N* and QTERM on the CardZ180.)

The problem installing the ZCPR module as part of the bootable system remained. Because the system was optimized for the TEI/Cray modifications, the CPMGEN.COM system relocator had been disabled (gave SYNCHRONIZATION ERROR after the first user response). I needed an image of the OS on which to install ZCPR. I remembered some things about SYSGEN.COM and decided to try it in reverse! I zeroed out memory with DDT and ran SYSGEN. After instructing SYSGEN to read the boot tracks of the standard boot floppy, I typed Control-C to warmboot with memory intact.

I SAVEd an appropriate-sized chunk of memory and found the start of the CCP in the memory image. It was then a simple matter of overlaying the version of ZCPR assembled for a BDOS location of C500h using DDT and SAVEing the new ZCPR64.COM. One SYSGEN later, I had a ZCPR boot disk. As I tend to do, I giggled hysterically when it came up the first time!

Continued on page 20

Regular Feature

Intermediate

The Mailbag

Dr. S-100

By Herb R. Johnson

Dr. S-100's fall column for September 1995. Herb Johnson, CN 5256 #105, Princeton NJ 08543.
hjohnson@pluto.njcc.com

Good news on the **GIDE** front, and the usual **mail**. Someday I'll get back to **hardware**, but **someone's** got to move this hard disk business forward. I've had a lot of interest in the **Jade bus card** and will write about it next issue.

GIDE IDE hard drive to Z80 interface

For those of you not familiar with GIDE, it is an IDE hard drive interface card that plugs into a Z80 socket. Tilmann Reh in Germany is the designer of this card, based on his work two years ago on a Z180 single-board computer with an IDE interface using a PAL (programmed array logic) chip for the logic "glue" to communicate with the drive. Several people, including myself, suggested he come up with an independent interface for any computer; I suggested he make it pluggable into a Z80 socket, and that he **add a clock chip** for a time of day clock. A year later, he now has some prototypes available and I am negotiating with him to import these into the United States. Software, of course, is not available: those who buy this prototypes must be able and willing to write Z80 drivers and share the code with others.

Here's the deal I have put together so far, not knowing all the final costs: If you are in the USA and are interested, AND can write CP/M BIOS code in assembler, you might want to work with this board. I'm taking orders in the USA at **$60 each without clock chip, $73 with**, delivered unassembled and subject to availability from Tilmann. You will get a board and parts, with whatever docs is available. If you want details, contact me (Herb Johnson) via mail, phone, or (preferably) e-mail. If you want **information** (two years of Tilmann's *TCJ* articles on IDE and earlier GIDE, and descriptions of the clock chip), **send $5**. Another $2 gets a disk of whatever software we have (mostly other people's software on doing various BIOS extensions).

I wish I could offer more and be definitive, but that's how it is today, at the end of August 1995. Remember, this is **development** stuff, not "plug and play", so if you can't write Z80 assembly language and don't know your own systems at the BIOS level, you will not find this useful. After some folks get these running and I can manufacture a regular supply in the USA, it may become more useful. Meanwhile, several people have previously given me deposits and are now ready to order.

More donations

I received a couple of **Compupro systems** from **Elliot C Payson** of **Denver CO** in July, the usual situation: "My goal is to clear out a public storage room by the end of this month". Elliot has been a customer of mine for some years, and contacted me when he decided to get out of the S-100 hobby. He very kindly shipped me cards, disks and docs from his two systems, and even took time to remove the motherboards from the boxes! I regret not being able to take the disk drives and the Compupro cabinets, particularly the latter. Compupro built heavy supplies with **regulated transformers** to insure reliable operation. Elliot said he'd hang on to the chassis and disk drives for awhile if anyone was interested, but that was until the end of July. I suggested he take them to a electronics surplus store we were both familiar with, as I lived near Denver for several years.

Anyway, his boards and such arrived intact, and I look forward to integrating his software with the system I wrote about in this column some months ago, also acquired in a similar fashion. Elliot had another version of an MS-DOS for the Compupro, which I hope to write up someday. Elliot's closing words were: "I had a lot of fun with the S-100 gear, getting it to work in varying degrees. Time doesn't seem to keep up with my project plans, so when our children grew up and moved away, we played more golf and traveled more, so a lot of unfinished projects remained that way for years. Now, after moving and less space, the CP/M gear has got to go."

GSX and graphics

I continue to correspond with **Emmanuel Roche** in France. Regular *TCJ* readers will remember his discussions of **Digital Research's GSX graphics standards** and he is also a correspondent with several CP/M enthusiasts. He writes in June: "Thank you very much for your parcel [a copy of David Cortesi's "CP/M Programmers Notebook" and some S-100 cards]. I was particularly impressed by his methodical way of programming and using his library of subroutines. The [Digital Research CP/M] MAC manual also promotes very much this, with the SEQUIO.LIB which was probably used in a few CP/M 2.2 programs, like UNLOAD vers 2.2. Thank you very much for the photocopies of manuals dealing with the NEX uPD7220 GDC [graphics chip, as used in the graphics cards in my

TCJ article months ago]. I found the CSD one very interesting."

"Thank you very much for the two 8-inch disks (of graphics routines from the same graphics companies). These files will be a nice addition to my disassembles of NCRGRAF (a package of graphics routines to be used with MBASIC), EGRAFPAK (some graphics routines in source code from EPSON for the QX-10) and the GSX screen drivers for the Epson QX-10, and the NCR DecisionMate V. It is quite interesting to compare these seven ways of using the same chip...."

I also sent him an S-100 graphics card, one of a dozen I have from a banking system, which uses the Motorola 6845 CRT controller chip, the same chip used in the original IBM-PC mono and CGA color cards. "Thank you for the M6845 card, but I must admit that I was puzzled that you sent it to me as, at first glance, it has nothing to do with the NEC uPD7220. I sent it to the "S-100 Expert" of the CP/M User's Group UK. Could you explain a little more how you got them, and why you sent me one?" Well, his interest in graphics cards suggested to me he'd be interested in one; and his abilities to decipher old software suggested he could decipher this undocumented card.

"Finally, I want to let you know clearly that I would be VERY HAPPY to buy from you the I.T. card with its monitor, once you no longer need it, as it would provide me with the most powerful GSX screen device under CP/M on a S-100 system. Digital Research wrote a GSX screen driver for the NCR DecisionMateV which was also using this chip. Patching the I/O port numbers and disabling some ROM checking code would produce a ready-to-use GSX driver. Then, the two GSX programs, DR GRAPH and DR DRAW would become usable. That's the big advantage of GSX: it really works with several devices. (But it is big and slow, so not usable for games.) You already saw [in a previous letter] what you can get from the HP plotter. Compare that with a screen dump on a dot-matrix printer!"

Well...like Elliot Payton, I got a lot to do myself. **I'd like** to get back to the Compupro system and the IT card. But Emmanuel will be first on my list to get this card. I had hoped the materials I sent, and the S-100 card in particular, would give him something to work on in the meantime. I noted that the OTHER graphics card in my Compupro system, with a TI 9940 graphics chip, would be easy to wire up by hand, and would offer reasonable graphics too!

CP/M on the Internet Cyber-cafe

Later in August, Emmanuel writes: "Please find enclosed the results of the investigation of the mysterious S-100 card you sent to me without explanation [the card with the Motorola chip], done by the S-100 expert of the (now defunct) CP/M User Group UK. I was recently able to connect to the Internet [again] via the first "Cybercafe" opened in Paris. It was interesting, but the price is too high in France - one dollar every five minutes. I saw the contents of the "comp.os.cpm" newsgroup, but was deceived by its small size (messages probably have a time limit duration) and not technical contents (just silly messages). I am thinking about launching one such newsgroup, with more technical content like the disassembly of the BIOS of the Amstrad PCW8256; as more than 1.25 MILLION were sold in Europe. [Meanwhile,] the last *TCJ* I got was March 1995 - I have no idea what is happening."

Emmanuel enclosed a letter from his UK colleague, **Ken Mackenzie of Luton, England** (if I read the address correctly!) which describes the video card I sent. Apparently the card supports 16-bit data transfers via DMA. He hypothesizes that it was built in the last days of the S-100 world to compete with the direct video speeds of the IBM PC. As it was built to support banking transaction teller terminals, that was a reasonable guess! He said he could not get the card working because of a lack of information on some of the chips, and the absence of a compatible connector to the "funny video" connector on the card. I had presumed Emmanuel would simply solder a proper connector onto the card. As for sample software, some old BIOS code from the IBM PC that supported the mono or color card would probably be a good start. I'll have to send some over in my reply...

Altairs: wanted and found

Sam Hevener of Richfield OH just started reading *TCJ* in June: "I just received a sample copy and read your story on page 36, and saw your ad on the inside back cover.

I was a mainframe computer repairman from 1967 until 1988, during which time the "PC" came out. In 1975 I was very much interested in the **Altair 8800 computer**. I purchased the instruction and assembly manual from MITS and was about to purchase the 8800 CPU. I still have those manuals today. What really turned me off on the "PC's" was the fact that the systems that came out just after the Altair did not use the S-100 bus. [The IBM PC was actually a previously rejected design for a terminal, and I don't think "compatibility" was IBM's strong suit in 1981!] Every manufacturer had his own idea of the "best bus". From that time on, I have not had any interest in "PC's". I don't like the idea of being locked into one manufacturer.

I would like to purchase a "dead" (non-operating) but complete Altair 8800 CPU, not the Altair 680 [a 6800-based product also by MITS], in good display condition. I have no desire to have an operating unit. Again the CPU must be complete but can have defective boards.

My hobby is collecting WWII military radios which means I'm used to the old electronic items. I just like the history behind many of these old items. If you don't have any of the units I describe, can you put me in contact with someone who does?"

I just did! But I have to say that there are many other people with just the same interest and motivation you have for an MITS/Altair 8800. I have "heard" that these go for over $1000 each, but I can't say someone has **told me** that is what they paid. So I'm afraid general interest in this machine is rather high. However,

IMSAI's are less well-known and so are more available. Furthermore, they are a better design, and more were built. And, there are other S-100 machines of similar vintage or later vintage that are easier to come by; and each manufacturer has its own "history" too.

As with all my correspondents, I encourage anyone interested to contact these folks and exchange information and technology. As for where to get these machines: try garage sales, hamfests, computerfests, auctions, newspaper ads, and so on. While I sell S-100 cards and docs, I find it is too difficult (i.e. expensive) to sell systems. Shipping, and the effort to provide a complete and working system, jacks up the price; whereas a card or two is easy to manage in costs and efforts.

R. O. Whitaker of Apollo Beach FL reminds us of the problems of inventing too early, as well as how Altairs were originally used. "I am a new subscriber to *TCJ*. Own a pair of original Altairs. They were purchased to be used in a demo system of a replacement for a newspaper, a system on which I hold the patent. The news is transmitted from a radio transmitter, each story only once. It is received in the home and stored in computer memory. Then whenever the user wishes to read the latest news he sits down and calls the selected stories from memory."

"I got a demo working using the two Altairs. But interest was nil or negative. Caught quite a bit of ridicule."

"I then set them up to demonstrate a base-16 numbering system using "Computer Compatible Numerals". That went over no better. Gave up on that several years ago. Since then they have been gathering dust. Plus mouse droppings."

"My chief objection to the S-100 bus was the scattering of the address lines hither and yon — rather than having them in succession in a line. I have one of the Jade bus probes you mentioned in an earlier article. It is in virtually mint condition. Should also have the literature on it. I would be willing to lend the probe to the person who was looking for one - he might get in touch with me. If I had had extensive use for the Jade probe I would have added an adapter which would have put the lines in order." [I would suppose that is what the front panel does - Herb]. I would be willing to write something up for you if you feel that I might have anything in which you would be interested."

Of course, you already have. Thanks! It might be interesting to see the bits of code you wrote to simulate the "radio newspaper" you put together, and how far you carried the simulation. Did you decode binary signals from a modem, or directly from audio? Did you compress text? and so on. Your patent would seem to be a neat bit of "prior art" that might keep some big software company from "patenting" certain kinds of software, but I'm no lawyer.

Systems found: now what?

Ron Wintriss of Lisbon NH writes: "I'm an avid reader of *TCJ*, but I have not needed your services and advice until now. At the local hamfest I ran across two S-100 systems I just couldn't pass up. I was guaranteed that they both worked, but upon getting these home I found some "problems".

"Computer number one is a **Northstar Horizon** with two 5.25 inch floppies and a Hazeltine 1500 terminal. Two disks came with it. I was assured these were the boot disks, and I took his word as gospel. On power up, the left drive started reading and a diagnostic program came up on the screen. Whatever menu item I chose, the results were the same: machine hangup. After trying and resetting five times, the program no longer comes up. Maybe I scared it. What I need to know is do these things use some kind of DOS? Are they like CP/M machines with software BIOS?"

Ron, the quick answer is that the Northstar was used as an independent computer, and also as a diagnostic machine for some mainframe computers, most notably the Cray. There were operating systems for the Northstar, namely CP/M and NorthStar DOS: I have these and I'll contact you for details.

Ron continues: "Machine number two is an **MITS Altair** with lots of boards, some manuals, but no Altair operation and programming manual. These I need. Also it is just a mass of boards in a backplane, i.e. no chassis or cabinet. Do you know where I can get a chassis at least for this unit"?

Depends on what model the Altair is: is it the original 8800 with the little round switches and blue front panel? Is it the later model with the flat switches and a black front panel? I'll contact you for details. And I have some manuals, but no cabinets to spare.

"I also read your column, and was surprised to see the Jade Buss Probe mentioned. My brother sent me one in the original box. I didn't know what I was going to do with it, but now I'm glad I kept it.!"

You might try putting it in the Northstar to observe it's operation. You can monitor what the computer is doing in response to disk access, terminal keyboard entry, etc.; and get some idea whether it's reading your disk or terminal. Read the manuals first, of course, before plugging anything in!

Karl Fritz of Akron OH is looking for SD Systems boards, and for Digital Research of Texas boards, namely 1 meg RamDisk cards.

References:

Elliot C Payson, 6325 W. Mansfield Ave, No 206, Denver CO, 80235
Emmanuel Roche, 8 Rue Herluison, 10000 Troyes, France
Ken Mackenzie, 116, Montrose Avenue, LUTON, Beds, LU3 1HS, England
Sam Hevener W8KBF, 3583 Everett Rd, Richfield, OH 44286-9723; (216) 659-3244
Karl Fritz, PO Box 9080, Akron, OH 44305.
R.O Whitaker, 128 St Kitts Way, Apollo Beach, FL 33572.
Ron Wintriss, 100 Highland Ave. Lisbon, NH 03585.

The European Beat

by Helmut Jungkunz

Regular Feature

All Users

East German Z80

Today: KC and the Sunshine

Once upon a time, computers were very expensive. A special 8-bit home computer named KC ranked around US$ 12000 !! Only, at the time you couldn't. And most likely, you wouldn't have bought it any way, since the time I am talking about is 1984-1985, when the term "East Germany" still meant a very different country. The KC was sort of Top-Of-The-Notch for the 8-bit developer.

What do I have to do with this? Well, this year, there were two German Z-fests, one in Eastern Germany, and one in Western Germany. Both were attended by Jay Sage and Joerg Linder.

Joerg Linder is a very active 8-bit hobbyist and one of the main cooperators of my ZNODE 51. He has produced quite valuable translations of program descriptions like QL41.DOK, NULU152.DOK, SLR.HLP and many, many others. At the moment, he is working on a translation of the documentation to BYE, since the BYE-concept is fairly unknown to most of the German 8-bitters. I found this situation quite unsatisfactory and asked Joerg to do some work on BYE, and he agreed, being unfamiliar with it himself. After all, Language barriers are amongst the main reasons for low interest in main concepts, which made it nearly impossible for me to persuade anybody into taking a dive into ZCPR, before the advent of NZ-COM and Z3PLUS.

So, to cut a long story short, Joerg was there at the Z-fest in Gueglingen, where he brought his favorite little machine, the fabulous KC-85/4.

I was so surprised about the elegant design, that I asked Joerg to supply me with some information, so that I could write this article.

The true makers of the KC series always saw themselves totally apart from VEB Robotron, the official marketer. The Robotron machines typically looked like Army surplus boxes, all in green, almost like home made weird-ware, with a totally incomprehensible keyboard. Not so the KCs. Compared to the Robotron stuff, they were really beauties.

First of all, there was the KC 85/1, already equipped with the famous U880 CPU. This was a processor, manufactures for export as well. The flawless ones carried the print "Z80", the less perfect ones "U880". One of the stories told, is, that one day Robotron had used up all existing U880s for the production of their computers and ran short all of a sudden. The Plan had malfunctioned again. What else could they do, but buy Z80s back from West Germany!

Technical data

——— KC 85/1 ———

CPU:	U880 D
Memory:	17K RAM
User Memory:	16K RAM
ROM:	6K ROM
RAM Expansion:	64K RAM maximum add-on
Keyboard:	65-Keys
Display:	TV-Set
Screen:	20x40 or 24x40
Characters:	8x8
Graphics:	128 symbol graphics
Color:	optional RGB
Sound:	Buzzer
Storage medium:	Cassette tape
Connectors:	TV, Cassette, special I/O
	2 Joystick-Connectors
Languages:	BASIC, Assembler

——— KC 85/2 ———

CPU:	U880 D
Memory:	32K RAM
User Memory:	18K RAM
ROM:	4K ROM
RAM Expansion:	4096K RAM maximum add-on
Keyboard:	64-Keys
Display:	TV-Set PAL!
Screen:	320x256 dots
Characters:	8x8
Graphics:	81920 dots
Color:	16 foreground, 8 background
Sound:	2 generators, 2x5 octaves
Storage medium:	Cassette tape
Connectors:	TV, Cassette, special I/O
	2 Joystick-Connectors
Languages:	BASIC, Assembler

Modular expansion modules available (PIO, serial etc.)

——— KC 85/3 ———

CPU:	U880 D
Memory:	32K RAM
User Memory:	18K RAM
ROM:	16K ROM
RAM Expansion:	4096K RAM maximum add-on
Keyboard:	64-Keys
Display:	TV-Set PAL!
Screen:	320x256 dots
Characters:	8x8
Graphics:	81920 dots
Color:	16 foreground, 8 background
Sound:	2 generators, 2x5 octaves
Storage medium:	Cassette tape
Connectors:	TV, Cassette, special I/O
	2 Joystick-Connectors
Languages:	BASIC from ROM!, Assembler

——— KC 85/4 ———

CPU:	U880 D
Memory:	64K RAM
User Memory:	64K RAM
ROM:	20K ROM
RAM Expansion:	4096K RAM maximum add-on
Keyboard:	64-Keys
Display:	TV-Set PAL!
Screen:	320x256 dots
Characters:	8x8
Graphics:	81920 dots
Color:	hires
Sound:	2 generators, 2x5 octaves
Storage medium:	Cassette tape
Connectors:	TV, Cassette, special I/O
	2 Joystick-Connectors
Languages:	BASIC, Assembler

KC 85/2, 3 and 4 can be chained with external boxes up to 16MB. A disk unit with CP/M compatible MicroDOS 2.6 (at 4MHz!).

Okay, now let's have Joerg tell us it's story:

Some history

Initially, the KC-series was conceived for the home market. This is the reason, that some KC 85/2 still display "HC 900" when coming up, HC as in home computer. Due to the shortage on the electronic supplies (especially Memory-chips were gold dust in the GDR!), decisions were made, only to produce these computers for official purposes.

The first computer from Muehlhausen was introduced in 1985 at the Leipziger Messe as KC 85/2. The reason for this strange version number /2 was the simultaneous release of a KC 85/1 from Robotron, both machines being completely incompatible. Shortly after, the meagerly equipped KC 85/2 underwent further development. The output was the KC 85/3. Even if the year 1987 had already come, the name was kept for the sales. The GDR ministry of education had finally realized the necessity of computer education. The chosen "instrument" was the KC 85/3, the reason why this computer was produced in larger numbers. The top-of-the-list product out of Muehlhausen was the KC 85/4. It went into production in 1989 and was even publicly available. If not for the "Wende", the reunion of both East- and West-Germany, there would have possibly been a far larger number of those computers.

One other product to be mentioned, is the KC-compact. Although being totally independent of the KC-series, it had it's importance in being compatible to the AMSTRAD CPC. They copied some of it in such a perfect way, that even the same ROM-bug can be found within the KC-compact and the CPC. Only a few of those computer exist, since the production started in mid 1989.

Some technical stuff

Basically, all KCs are of the same construction. The housing is about 385x270x77 mms and contains the power supply, the main CPU-board and two module slots (actually more like connector arrays to the rear, H.J.). The KC was equipped with a not so comfortable keyboard in a separate housing. Standard video output is through a TV-antenna, but a better picture can be achieved by the rear RGB-connector (supposedly analog).

Inside the KC, a U 880 CPU operates at 1.75 MHz. This CPU is fully compatible to the Z80 (see above <G>). Since the CPU has to cope with the color graphics all by itself, in some respects, the performance is rather slow. Nevertheless, the modular concept of the system is quite pleasing. Using a clever module management, and through the connection possibilities of expansion add-ons, the system can be expanded nearly limitlessly.

The address room of 64K is kept in 16K blocks. The KC 85/2 comes with 16k RAM on delivery, 16K screen memory and 8K ROM. The KC 85/3 has an added internal BASIC-interpreter, so it uses 16K ROM. In the KC 85/4, there are 64K RAM Memory, 64K screen memory RAM and 20K ROM.

There were 35 different modules, including various test and measuring modules, not all of them having ever been available in the stores. They can be basically divided into three groups: ROM-modules (even with software), RAM modules, and I/O-modules. The modules are governed by a PIO, that is used like a MMU. By that, it is theoretically possible, to address up to 16MB of memory.

You could get a bus driver and an intelligent floppy disc substation. Each bus driver will allow you to drive four modules. By far more interesting is the "Floppy Basis". It contains a complete Z80-computer, running at 4 MHz. Four drives with 780K capacity can be hooked up to it.

Lucky owners of said Floppy Basis can run two operating systems. One, the KC operating system is expanded to disc access, second, another system called "MicroDOS" can be run.

Some software business

MicroDOS is compatible to CP/M 2.2. A very unusual version number of 2.6 is reported. The basic CP/M 2.2 has been added a few functions (like SCB), so it ranges somewhere between 2.2 and 3.0. Thus, only about 50K TPA are left. But while integrating Tilmann Reh's GIDE by our software specialists, a more powerful operating system will (have to) result.

Also very interesting is the interaction of both computers. The actual KC is mainly functioning as terminal under MicroDOS, while the Floppy Basis does the real work. All data transfer is executed via coupling RAM of 1K size, situated within the address range of the Floppy Basis, as well as within some of the I/O-range of the main KC.

The advantage of such a construction design, is the usability of all features of the basic device within MicroDOS. For instance, color graphics can be used, which on the other hand would make these programs incompatible with other CP/M-machines. Also, all connected RAM-modules are recognized and accessible as one RAM-disk. This RAM-disk can be up to several MBs large.

Some prices

I am certain, these prices will be interesting to everyone:

Since the KC 85/2 and KC 85/3 never had been available to the public, the prices are mere estimations. In the GDR, companies had to pay higher prices than ordinary people. A PC 1715 with green monitor and 9-needle printer was 20.000,— to 25.000,— Marks (East). Don't forget - we are looking at a Z80 computer with CP/M 2.2 after 1985!!

When the KC 85/4 first appeared in the shops, it's price was 4.500,— Marks (about 4 times the average monthly income .).

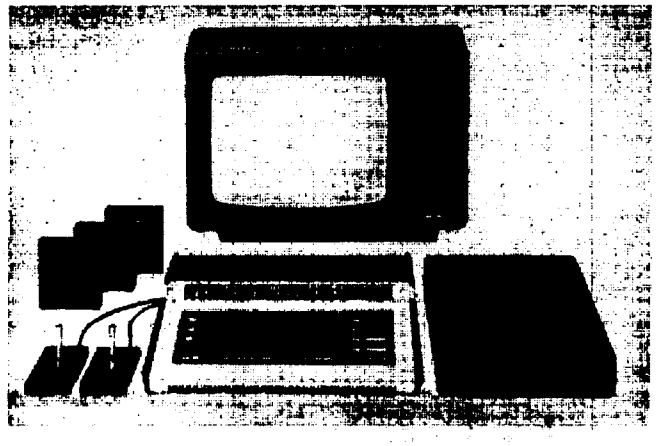


Bild 7.1.1. Minimalkonfiguration des KC 85/1 ergänzt durch zwei Spielhebel

A 16K RAM module rated more than 750 Marks. But despite the exorbitant prices, computers and spare parts were hard to get, as digital electronic components were scarce, as they were usually exported into the NSW (non-socialist-economy =the West).

A small glance into the future

Since the manufacturer is defunct, we must help ourselves and each other. This was the reason to form the KC-Club in 1992. The initial founder had given up after about 1 year and things had become quiet about our club. So, I took up the lead. Ever since October 1994, the KC-Club exists, and with it a magazine.

In the mean-time, the number of members has increased to a number 70, of which a greater part is quite active. In the middle of April 1995, we had our first(!) club meeting. It was a big success- 30 members showed up. The next meeting is already being prepared.

Our projects at this moment are: connecting Tilmann Reh's GIDE, plus a more comfortable keyboard, development of a sound module, speeding up the KC, development of a better MicroDOS. These are only the larger projects. Many of our members are building a few things or writing smaller programs, that I cannot list here altogether.

I hope to have brought you some interesting insight into the world of the "KC".

Bye, Joerg Linder (translated by Helmut Jungkunz)

I'd like to thank Joerg for this "essay" on the amazing world of GDR. Hope you found this as fascinating as it was for me. Regards and cu; Helmut Jungkunz.

(P.S: I'd be glad to receive your comments on my articles, best through e-mail, since this creates a computer file that can be passed to others.)

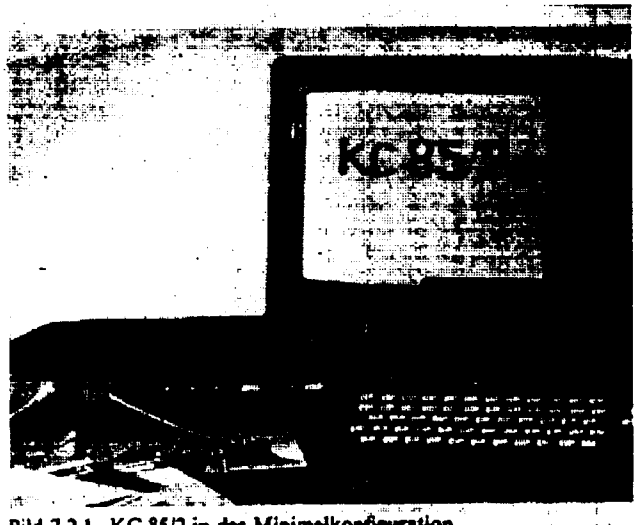


Bild 7.2.1. KC 85/2 in der Minimalkonfiguration

32-Bit Systems

All Readers

Rick Moved

Real Computing

By Rick Rodman

Welcome to the newly-established Kettle Pond Computing Facility. In addition to a number of PCs and a grand total of almost 8 GB of online storage, there is a number of experimental and vintage computers for software testing and portation.

Real Computing is not just an attitude, it's a way of life. We take our computing seriously here. We believe not just in fast computers and low overhead, but in squeezing the last drop of performance from our hardware. While many folks are OLE-embedding Microsoft Word so they can print an envelope from an interpreted Visual Basic program, we shudder at such waste.

Real Computing is like hot rodding. Hot rodders know their hardware, whether it's a 1936 Ford Coupe with a supercharged V-8, an Austin Mini-Cooper with a transverse 4, or an old Saab with its 2-cycle, 3-cylinder motor. It doesn't matter how powerful your engine is - what makes the difference is how well you understand and use that engine.

What I'm getting to is that I've moved, and with everything in chaos at time of writing, I haven't gotten much computing done lately, but I have positioned myself for better computing in the near future. I have played with the Minix 1.7 beta, but have experienced problems with it booting on two machines I've tried it on. It is reported to include TCP/IP and support for Adaptec SCSI boards. Minix may become my universal OS - running on experimental hardware - or I may go back to work on Uzi.

Latest news on Uzi

I've heard rumors that some folks in Germany worked further on Uzi, but have been unable to confirm these. However, I have had some e-mail exchanges with Doug Braun, the original author of Uzi and Uzi-280, and he basically indicates he has no further interest in Uzi and has not worked on it in some time.

Because it is so compact, Uzi is a possible candidate for our portable operating system. The idea here would be to use it for small embedded systems based on Z-80 or other processors, and as a sub-operating system on our development platforms. Rather than taking over the machine, it would run within MS-DOS or CP/M, much like Z-System does.

On the target machine, Uzi would run 'native', providing a Unix-compatible environment to programs.

Since Uzi is less complex than Minix, it could be made easier to port.The scheduler needs work, however. Another thing which needs to change is that programs need to be loadable in relocatable format rather than COM format - SPR or PRL, for example.

Oberon

Another interesting "free" package is Oberon, which is a combined operating system and language. The language is similar to Object Pascal. However, rather than having a rigid boundary between the operating system and programs that run within it, a programmer writes functions which are added to the system and add functionality - much like APL or Forth. The operating system and compiler are themselves written in Oberon. There are a number of different versions available, and it's not clear how much source code you get. Nor is it clear how efficient or portable the system is. It doesn't come on CD or diskette; you have to ftp it from Switzerland over the Internet. Because the files were compressed in 16-bit Unix compress, I haven't been able to unpack them or try it out yet, but if it looks good, I'll have a full report.

Sprite

I've talked about Sprite previously, but in response to reader inquiries, I have a little more to say about it. Sprite is a distributed OS with almost-full source available from Walnut Creek CD-ROM. It appears to include a bootable image for the DEC 5000 workstation and for a Sparcstation 2. I can't tell whether the Sparc image would work on my Sparcstation 1 or not.

Sprite is full of really clever ideas, such as the Log File System, where everything is written to a file as a stream of changes. There are a number of papers on the CD, such as a comparison of Sprite with Amoeba. And, of course, there is an astonishing amount of source code on the disk, although some source, being AT&T-encumbered, is missing.

There is probably too much code, in fact, for a hobbyist to hope to do more than mine it for useful functions and modules. It might be possible to use "chunks" of Sprite in developing your own distributed operating system; however, in the past I've had little success with this.

The reason is that the module granularity is too large.

I'll digress for a moment and explain what I mean by this. Visualize each source file as a little block floating in space. Then, you can see inter-file connections such as global variables and function calls as little glowing threads between the little blocks.

In a small program, there will be a few blocks all connected together with hundreds or even thousands of threads. To try to remove a block and use it in a new program, the new program will have to satisfy all of those little connections.

In an operating system, there will normally be "clusters" of tightly-interconnected blocks connected by comparatively fewer threads. Each cluster can be considered a module. For example, in Minix, you would have the FS and MM modules floating off, with very few connections to them. You can consider programs which run under Minix, such as the shell, to be connected through the system call interface. In another example, under CP/M, the CCP, BDOS, BIOS, and transient programs, are connected to each other by only one or two threads.

Sprite has a modular design consisting of about 30 modules. Because it's a distributed operating system, I would expect that the modularity may be better than in most other operating systems. Each of the modules is very large, though; for example, LFS contains 43 source files. The code is well-commented. So, if you're writing a big operating system and need a source of ideas, either from research papers or from source code, Sprite may be of value to you - possibly greater value than other, less modular systems like FreeBSD or Linux.

Personally, I think the modularity is still too coarse. Modules should be much smaller, and protected from each other, if possible, by hardware. I'd like to see a Minix-size operating system with thirty or forty very small modules instead of three.

The goal of a modular design is to tightly specify intermodule connections so that modules can be easily moved from one project to another. The number of intermodule connections must be reduced to a minimum. Then, each module can be easily reused in other programs. It's desirable that modules be as small as possible, so they are functionally separable, not just agglutinations of convenience.

Sounds quite simple, doesn't it? Unfortunately, almost nobody writes programs that way. Somebody will undoubtedly say, "Object-oriented programming!" But OOP, as presently practiced, *prevents* modularity because of inheritance. By creating a hierarchy of objects inheriting from one another, an OOP programmer creates modules which are tightly interconnected by a large number of impenetrable bonds. Only very simple objects could be extricated from a project written in purist OOP fashion. While presenting an illusion of encapsulation, OOP has merely hidden the spaghetti beneath the meatballs.

Miscellaneous news

In unrelated news, I've been playing with a CD called "Fun in the Sun" from the Sun User's Group. There's a lot of neat noises and some X Window games on it, most of which I've seen before. I've also acquired some "Prime Time Freeware" CDs at giveaway prices at Micro Center, who is closing them out. They're chock full of source code.

The deluge of 286 PCs has begun. Most XTs have by now either found new homes or wound up in landfills. But there were at least ten times as many 286 PCs made. Expect to see truckloads of them at hamfests and flea markets, with prices dropping into the $10-$20 vicinity. These machines are much easier to configure than XTs, especially adding hard drives; although the CMOS batteries will probably be dead and no technical information available, they'll make nice wordprocessor machines for your kids or favorite charity. They won't run Linux or Windows 95, but they'll run Minix just fine!

Next time

Next time we'll have a review of FreeBSD 2.0, the other "free" Unix. This version comes with a recommendation from Walnut Creek CD-ROM, who run their entire operation using it. FreeBSD is closely related to the NetBSD release which many PC-532 users have switched to.

Reference

Real Computing BBS or Fax: +1 703 759 1169
E-mail: 102046,1656 on Compuserve (102046.1656@compuserve.com)
Mail: 1150 Kettle Pond Lane, Great Falls VA 22066

Walnut Creek CD-ROM
1547 Palos Verdes Mall Suite 260, Walnut Creek CA 94596
+1 510 675 0783

LINUX $61.95
Slackware Pro 2.2
Includes 4 CD-ROMs and a 640 page manual
A ready-to-run multitasking Unix clone for 386 and higher PC compatibles. TCP/IP, C, C++, X Window, complete Source Code, and much more!

JUST COMPUTERS!
(800) 800-1648
Fax (707) 586-5606 Int'l (707) 586-5600
P.O. Box 751414, Petaluma, CA 94975-1414
E-mail: sales@justcomp.com
Visa/MC/Int'l Orders Gladly Accepted
For catalog, send e-mail to: info@justcomp.com
Include "help" on a single line in the message.

Regular Feature

68xx/68xxx Support

C & Assembly

Small System Support

By Ronald W. Anderson

C Tutorial #4

In order for a program to do much useful work it has to be able to get information in, process the information, and output the results. This lesson will concern itself with input from the terminal and output to the terminal. Later we will look at input and output from files, which closely parallels I/O from the console or terminal.

Before we can get into using these functions, however, we need to talk about functions and parameter passing. Let's build on what we learned the last two times and define a function that does nothing but kill time. That is, a delay function.

```
delay()
        {
        int n;
        n = 30000;
        while(n—);
        }
```

The function simply counts down from 30000 to zero and returns to the line after the program line that "called" it. The calling is accomplished by including a line with the statement:

```
delay();
```

Of course it would be nicer if we could control the length of the delay somehow. We can, fortunately. There is a mechanism for the main program to pass information to the delay() function.

```
delay(count)
int count;
        {
                while(count—);
        }
```

In this example the (count) part is the definition of a "formal parameter". It simply says that the part of the program that calls this function is going to "pass" it an integer number that is given the name "count" locally to this function. Function calls in other parts of the program will look like this:

```
delay(6);
delay(25000);
delay(timeval);  (assuming timeval is an integer variable)
```

For any simple variable type, the program "passes" the value of the variable to the function. This is called "passing by value". If the formal parameter is an array, however, it would be a lot of work and take a lot of time to pass the function all the values in the array, so C automatically passes the function the address (physical memory address in the computer) of the first element (item) of the array. Thus in our first session:

```
char message[] = { "Hi there" }

...

puts(message);
```

The call to puts passed the address of the memory location containing the "H" to the function puts(). puts() is defined in stdio.h so that it expects to receive an address, not a value. We'll expand on this next time when we get to the subject of pointers.

Let's take a moment to talk about "initializers".

int n; declares an integer variable n and does nothing to give it a value. It simply sets aside space for the variable. It may well contain garbage.

int n=0; declares the variable and gives it an initial value of zero. We could just as well do it with two statements:

```
int n;
n = 0;
```

The same is true of the array. We could define the array:

```
char message[17];
```

We would then have to use one of the string functions that we haven't talked about yet to put the message in the array, or we would assign the characters one at a time, (a real chore).

When in C we use a literal string as in:

```
puts("Hi there");
```

C treats the string just as though it were an array elsewhere. In fact most C compilers put all the literal strings together somewhere in memory as a group, and compile the address of the

literal string in place of the string itself. C also automatically ends a string with a NULL (0) value. That is, the compiler appends or adds a 0 to the end of the string. This can sometimes cause a problem if you are not careful. If you declare an array: char message[12] you can only put 11 characters into it because the compiler uses one location for the NULL that it appends.

Now having gone through that we are prepared to look at the printf() function. Printf() is rather unique in that it allows you to pass it a variable number of parameters. So far we have only used one parameter, but functions can use several. The simplest use of printf() is to print a string:

```
printf("Hi there\n");
```

This does just what you think. The only thing new is the escape sequence "\n". C has defined a number of these to print "non-printable" codes. The \n means "newline" and it outputs a CR and an LF to the monitor to get you to the start of the next line. Printf() can do far more than that, however.

```
printf("%d %c %x\n",'A','A','A');
```

result: 65 A 41

The format string in quotes has told printf to print three values as decimal integer (%d), character (%c), and hexadecimal (%x). The values are listed as the character constant capital A three times. The printout converts them to decimal 65, character A, and hexadecimal 41, all precisely the same value:

01000001 in binary.

Note that the spaces in the format string cause the spaces between the three values that are printed out. If the format string were "%d%c%x\n", the printout would be: 65A41!

I have included a list of the format specifiers. Things can get more complex, and do, but once you grasp the basic idea, output of numbers in different formats is easy in C.

Scanf() is the corresponding input function to get an input from the terminal.

```
scanf("%d",&n);
```

This gets the value of the variable n. Note one new idea here that causes confusion to new C programmers. The function scanf() doesn't want the name of the variable to which you are going to assign the input value. It wants the address of the variable. The ampersand "&" is the "address of" operator. The statement says to get an input from the keyboard, translate the sequence of ASCII characters into an integer and place it in the variable n.

I wondered frequently why scanf() was written this way. Couldn't you just as well have scanf() return a value of the type of the format specifier? Then you could simply say: n= scanf("%d"); While that is all true, scanf() allows you to input more than one value at once. Doing it the "couldn't you" way would only allow inputting one value with each scanf() call. We haven't talked about functions returning values, but as we will see later, a function can directly return only a single value. Passing a function addresses of variables allows it to return multiple values via the variables whose addresses were passed to it.

```
printf("input two integers: ");
scanf(" %d %d),&m,&n);
```

I think that will be enough for this session. Next time we will introduce pointers and show how they relate to arrays. Let me say that this series is intended mostly to act as an introduction to C. It doesn't take the place of writing some programs.

I am going to recommend a shareware program. If you have access to a PC, you ought to get this one. It consists of lessons in C programming presented nicely. If you have a C compiler you can link it to the program in such a way that you can compile some example programs with it. If not, you can learn C anyway, and perhaps use your 6809 system to work through the example programs in the lessons.

The package is called CTUTOR. It is sold by KNOWWARE, 1039 Melrose St., Bowling Green, OH 93402. The supplier wants a $10 registration fee and $3 for a 60 page printed manual. Registration assumes you already have a copy of the program. You can get a copy of the latest version for an additional $10, but it IS shareware and I can give you a copy. I have the latest version and I would be happy to do so. If you want to try it out, send me a formatted disk (5.25 or 3.5, format of your choice (for a PC, of course), and I'll send you a copy free.

If you like it and decide to register, there is information along with the program. Again I would advise starting at the beginning and going as far as you can (until you get lost), then next session start over again. Sooner or later, drop the first lessons and add new ones at the end. It won't take long if you work through the examples and play "what's wrong with this program", since there are numerous examples of common errors that you can try to find, and then the tutorial will point them out to you. I hope this series can act as an "ice breaker" so you won't have to start with a tutorial or a textbook and have it all be totally new to you. This tutorial program is not quite complete, but it goes pretty far. Unfortunately it uses the style of ANSI C, but the differences are not large.

Table 1 — Format Specifiers

Data type	Scanf	Printf
Single character	%c	%c
Signed integer	%d	%d
Signed double or float sci format	%e	%e
Signed double or float in dec.	%f	%f

Octal (letter o not 0)	%o	%o
Character String	%s	%s
Unsigned decimal integer	%u	%u
Hexadecimal (%X for uppercase)	%x	%x
Print percent symbol		%%
Decimal integer 6 digits wide field		%6d
Floating point, 6 digits wide field		%6f
Floating point 2 digits after dec. pt.		%.2f
Floating point 6 wide 2 after dec. pt.		%6.2f

Size modifiers

Short int	(8 bits)	%hd	
Long int	(32 bits)	%ld	%ld
Double	(64 bit float)	%lf	%lf
Long Double	(80 bit float)	%Lf	%Lf

Assembler

Last time I talked about writing a program to load and run position independent programs, that is, programs written in position independent code. I had a few struggles simply because I forgot a couple of lines of code, but in an hour or so of debug I found my omission and the program worked. The program is called LOGO (i.e. LOad and GO), and the syntax after you have copied it to your system disk as LOGO.CMD, is:

LOGO LOAD ADDRESS FILENAME PARAMETERS

I have modified LIST1 to be position independent. I'll present the listing below. For the moment assume the new program called RLIST1 resides on the working drive (1) in the form of a .BIN file. The command to list its source file would be:

LOGO 3000 1.RLIST1.BIN RLIST1.TXT

The load offset is 3000 (hexadecimal). Since the program has no ORG statement, it defaults to zero. Our load offset is added to the load address and it loads at 3000. The command line parameter for RLIST1 is RLIST1.TXT, so it lists that file to the terminal. Here is the program listing. we'll explain the new ideas below:

```
* PROGRAM TO LOAD AND RUN A POSITION
* INDEPENDENT FILE

NAM LOGO

*SYSTEM EQUATES

LOAD    EQU  $CD30
GETFIL  EQU  $CD2D
GETHEX  EQU  $CD42
RPTERR  EQU  $CD3F
SYSFCB  EQU  $C840
LOADOF  EQU  $CC1B
FMS     EQU  $D406
WARMS   EQU  $CD03
FOPENR  EQU  $01
FCLOSE  EQU  $04
EOF     EQU  $08

        ORG  $C100

START   JSR  GETHEX     GET THE LOAD ADDRESS
        STX  LOADAD
        LDX  #SYSFCB
        JSR  GETFIL     GET THE FILENAME
        LDA  #FOPENR
        STA  0,X
        JSR  FMS        OPEN FOR READ
        BNE  ERROR
        LDA  #$FF
        STA  59,X       MAKE IT A BINARY READ
        LDD  LOADAD
        STD  LOADOF
        JSR  LOAD
        LDX  #SYSFCB
        LDA  #FCLOSE
        STA  0,X
        JSR  FMS        CLOSE THE LOGO FILE

        JMP  [LOADAD]

ERROR   JSR  RPTERR
        JMP  WARMS

LOADAD  RMB  2

        END  START
```

There are several FLEX routines that handle command line parameters. One of them is GETHEX. If you have the programmer's guide you will see that GETHEX gets a hexadecimal number on the command line (without the dollar sign) and returns it in the X register. Since the first parameter is the load address we get it and store it away in a variable. We haven't used a variable for the last few installments, but you will note that LOADAD is defined with an RMB 2 at the end of the program. Variables can be placed before or after (or in some places in the middle of) the main body of a program. It really doesn't matter. At any rate we get the hex value and save it with the STX LOADAD instruction. Then as we did the past couple of times we get the filename into the FCB using GETFIL. If you look at the discussion of the LOAD routine you will see that we have to use a location referred to as the system FCB. That is the File Control Block used by FLEX to open utility programs etc. It is at address $C800 so we define it with an equate and we use it for our working File Control Block.

Now we open the file for read and set the binary flag as we discussed previously in talking about our copy program. This is particularly important because the file we have opened is a binary executable file. We go and get LOADAD and store it in a location called LOADOF in FLEX. If you look at the FLEX variable list in the programmer's guide you will see it at $CC1B. The LOAD routine uses this address to load the file. Actually it adds this LOADOF value to the load address of the program file, which if there is no ORG statement is $0000, so it amounts to being the load address. Again we have checked for errors in opening the binary file and simply exited using RPTERR to tell us what's wrong if an error is detected.

The last instruction uses an addressing mode we haven't used before. It is called "indirect" addressing. The instruction is JMP [LOADAD]. This tells the processor to jump (set the program counter) to the address held in the variable LOADADD, in this case $3000. JMP LOADAD would be an instruction to begin executing code at the label LOADAD. JMP [LOADAD] is an instruction to begin executing code at the address that is the contents of the variable LOADAD. In the parlance of C (we haven't gotten to that part just yet) we use a pointer. LOADAD's contents "point at" the place where we want to begin execution.

This works fine as long as the first thing in our position independent code module is the first executable statement in the program. In the case of RLIST1 that is so. Last time we modified MYCOPY to be RMYCOPY, and we made it position independent also. Note that LOGO is NOT position independent. It loads in the FLEX utility command space and then it can load a position independent executable file anywhere in memory. You don't have to try very hard to load a program right over FLEX in memory and crash the system, so you have to have some idea of what you are doing.

Safe load addresses for a short program like RLIST1 would be anywhere from $0000 to about $B800, depending a bit of whether you have some printer drivers or whatever tucked away at the end of memory. Type the command MEMEND to see where the end of user memory is, and then back up about $200 bytes from there as a safe maximum load address for RLIST1.

After you have digested LOGO a bit, type it in and assemble it and then modify LIST1 and try the above command line. After you are done running it, push the reset button on your computer and examine memory where you said execution should begin. I.e. if you loaded the program at $3000 type e 3000-30ff and watch the monitor dump that section of memory. If it is not obviously the RLIST1 program, go assemble that with an output listing and look for those codes at address $3000. Now repeat the process loading the program at $2000 or whatever.

By the way, here is RLIST1:

```
* PROGRAM TO LIST A FILE TO THE SCREEN
* GETS FILENAME FROM COMMAND LINE PARAMETER
* THIS VERSION IS POSITION INDEPENDENT

NAM LIST

PUTCHR  EQU  $CD18
WARMS   EQU  $CD03
RPTERR  EQU  $CD3F
FMS     EQU  $D406
FMSCLS  EQU  $D403
NXTCH   EQU  $CD27
SETEXT  EQU  $CD33
WRKDRV  EQU  $CC0C
PCRLF   EQU  $CD24
GETFIL  EQU  $CD2D
CR      EQU  $0D
FOPENR  EQU  $01
FCLOSE  EQU  $04
EOF     EQU  $08

* CODE TO GET FILENAME FROM COMMAND LINE
* THIS VERSION USES FLEX GETFIL

START   LEAX  FCB,PCR   THIS IS POSITION INDEPEN-
DENT
        JSR   GETFIL
        LDA   #1
        JSR   SETEXT    SETS EXTN TO .TXT IF NONE
SPECIFIED
        LDA   #FOPENR   OPEN FOR READ CODE
        STA   0,X
        LDA   WRKDRV
        STA   3,X
        JSR   FMS
        BNE   ERROR     FMS SETS NON ZERO ON ER-
ROR
LOOP    JSR   FMS        READ A CHARACTER
        BNE   ERROR
        JSR   PUTCHR    WRITE IT TO SCREEN
        CMPA  #CR       IS IT A CR?
        BNE   LOOP      IF NOT, OK
        JSR   PCRLF
        BRA   LOOP      GO AROUND AGAIN

ERROR   LDB   1,X
        CMPB  #EOF      TEST FOR END OF FILE
        BEQ   DONE
        JSR   RPTERR    TELL USER WHICH ERROR
*                       - X POINTING AT FCB
        JSR   FMSCLS    CLEAN UP BY CLOSING ALL
OPEN FILES ON ERROR
        JMP   WARMS

DONE    LDA   #FCLOSE   BRANCH HERE ON EOF
        STA   0,X       CLOSE THE FILE
        JSR   FMS
        JMP   WARMS

* NOTE NO ORG STATEMENT FOR THE FCB.
* IT JUST FOLLOWS THE END OF THE CODE AND
* PRECEEDS THE END STATEMENT.

FCB     FCB   0,0,0,1
        FCB   0,0,0,0,0,0,0,0,0,0,0 ELEVEN ZEROS
*              FOR FILENAME AREA
        RMB   305

        END   START
```

General

I've observed the past few times that writing tutorials on Assembler and C at the same time seems to bring about a large amount of "synergy". We talk about pointers in Assembler this time and we will talk about pointers in C next time, for example. I think this is largely because C is in one sense a high level Assembler. It is basically an assembler that can be the same regardless of the processor on which it is running. That

is, it hides the details of the processor, the registers, names, etc., but it still lets you program at that level. Of course the conveniences are there too. You can program at a higher level, much as we have done using system calls to FLEX in the 6809 assembler programs. Here again, C has defined a set of system calls of its own. They are again not only processor independent but operating system independent as well. No matter (for all practical purposes) the details of opening files at the operating system level. C hides the details and has a "standard" Operating System interface. You open a file for read on an old 6809 system just the same way as you open a file for read on a 70 Megahertz Pentium system.

Looking at it in that light, C is a standardized assembler and it has a standardized operating system interface. I'm quite sure that was no accident! Learn 6809 Assembler on a FLEX system and you can program a 6809 in Assembler in the FLEX environment. Learn C and you can program a computer based on any processor running most any operating system. To be sure it is not quite that simple because some systems allow longer filenames etc. but basically the statement is true.

I am NOT trying to say that learning to program in assembler is a waste of time, just that if you want to program several processors in Assembler, you have to learn the details of each processor and its associated assembler. It is most handy to be able to insert some assembler code in a C or other high level language program. Sometimes you can greatly speed up a program's execution by doing so.

I began my programming experience by learning to program a 6502 in machine code. (That basically means that I played the part of the assembler and translated the mnemonics into hexadecimal machine code instructions). I moved on to a 6800 and then 6809.

About the time of my first experience with the 6800, I learned how to use the Motorola Assembler editor combination called CORES (not as in ferrite memory cores, but as in CO-RESident, since the package had a combination (co-resident) assembler and editor. Several years ago, I did some 68000 programming in assembler.

I can say that each new processor you tackle becomes a bit easier than the previous one, though it becomes difficult to keep them all separate in your mind. It is a bit like driving three different cars having an automatic transmission, a three speed manual on the steering column and a "four on the floor" respectively. I have an automatic on the steering column and an automatic on the floor (or rather the console). When I switch cars I am always reaching for the wrong shift lever. I find, though, that once I have "shifted gears" mentally, I have no problem driving either one. I can say the same for writing programs in Assembler for the 6809 or for the 68000.

I have recently begun to do some simple Assembler routines for the 80X86. I'll probably not become as proficient in that area as in the 68 processors since the 86 machines have so much memory and run so fast that it makes little sense to struggle doing anything complex in Assembler when C is available and the C versions run just as fast as the assembler versions. I don't always care whether I use 8 bytes or 8K for a program anymore.

Reader to Reader, continued from page 7

Well, that's pretty-much how I spent my summer outside of taking a couple of classes in summer school. I must admit that I spend the majority of my time these days in Amiga-land. Due to the limited space in my apartment, I decided that my Amiga and CDTV would be my principle form of automata. For everything which _must_ be printed I'm using Georg Hessmann's PasTeX v1.3. About a year ago, I effectively beat the rest of Texas A & M University (students, at least) to the punch when it came to Web-surfing from the dorm or apartment.

Thanks to a package called DNet (for Dillon-Net after Matt Dillon of the Software Distillery) I was able to run Mosaic (among other things) long before the campus PPP/SLIP servers were brought on line. DNet software on a SunOS or Solaris machine and on the Amiga form their own network protocol between the Amiga and the host machine and an array of servers then make TCP/IP connections from the host machine to other machines.

However, DNet interest and development were already waning when I found out about it. AmiTCP had been released and was taking over. I couldn't yet justify installing AmiTCP since there's only one place I can use it. Instead, near the end of the spring semester, I learned about MultiLink (or MLink) by Ezra Story. MLink is a kind of TCP/IP faker that emulates both the AmiTCP and AS225 interfaces on the Amiga end and passes the function call to the TCP/IP protocol stack on a remote host machine running the UNIX half of MLink. (Kind of like DNet, but using applications expecting AmiTCP or AS225.)

After playing with the OS/2 IAK on my brother's machine, I think I'll go back and start working up an AmiTCP installation. There's only one place, now, that makes using MLink necessary, but that site runs SCO OpenServer and MLink won't compile/link under SCO yet.

Well, there you have it in a nutshell. Pretty big nut, though. Maybe we can push the issue to 64 pages? <grin>

Take care.
John D. Baker ->A TransWarp'802'd Apple //e CardZ180 Z-System nut //Internet: jdb8042@tam2000.tamu.edu, http://tam2000.tamu.edu/~jdb8042/

Continues on Page 44

Embedded Control Using The STD BUS

By Bill Kibler

Special Feature

Embedded Support

Details of STD BUS

There are many ways to do embedded control operations. It may also help to explain something about what embedded control is about. The main reason for all this is explaining where the Standard Bus fits into the picture.

INTRODUCTION

The term embedded control has become the buzz word of industrial controllers. These computers, that are dedicated to controlling things that perform work, have been around for some time. Almost all early versions were custom built for the application to be controled. The first ones were built to control lathes and milling machines. The object was getting parts to be built the same every time. Humans have problems making the same part the same way every time with the same accuracy, typically .001 inches of tolerance.

Setting up the machines in the old days, usually amounted to using some form of "JIG" or "PATTERN" for the operator to follow. With computers the jig or pattern is contained in a program that runs the machine. The operators job is to make sure it is done correctly and stays in specification. The early computers had lots of electronics in many cabinets, and yet in computing power it was practically none.

The earliest machines I worked on were simply AND, NOR, OR circuits arranged in ways to make program decisions. Later FLIP-FLOPS and other more advanced logic was added. Giddings and Lewis makes large lathes and for their first actual computer control, they got Motorola to allow them to put the 6800 CPU design in discrete logic on a single printed circuit board. The design was for 16 bit operation, all on a two foot square board.

As you can see these early computers used lots of real estate (very large printed circuit boards (PCB's)) and required multiple boards. Typically a CPU, MEMORY, INPUTS, OUTPUTS. The inputs were more than just single point status that might tell if the horizontal cutting tool had travel too far down the lathe. The machines usually had some fancy way of knowing exactly where the cutting tool was to within .001 inches. These indicators are called synchro and servos and determine small phase differences between two sets of signals to know where they are.

The first buses

The first buses were designed by each company for their own products. The problem with this approach - no compatible boards, so you had to make them all yourself. At first that enabled you to one up your competition until things started getting more complex. The more complex and computer like the longer the development cycle and chances of loosing your industry position. As CPUs moved form boards to chips and the whole computer was possible to be on one reasonable sized board, the question then became, why build the computer at all. Simply build only what could not be bought.

Both Motorola and Intel were producing their own BUS systems for industry to use. These BUSes were controlled by the handshake signals the CPU used. Intel's was the Multibus, while Motorola had the Exorcisor Bus (8 bit) and the Versabus (16/32 bit). Both have since upgraded these buses and added piggy back boards for even more enhancement. They however are the high cost or Cadillacs of the industry. They are also large and only BIG companies with BIG projects use them.

While Intel and Motorola had the idea big is better, others thought differently. Mostek and Pro-Log got together and developed the STD or Standard Bus. The idea was small building blocks for small jobs. Only use the exact amount of hardware for the project at hand. About 30 other vendors agreed with this concept which has kept it very much alive to this day.

A BUS

For industrial control, there is nothing special about using a BUS. Early S-100's found themselves in many industrial situations. I sold Teletek systems to a cloth manufacture for their weaving systems. In fact some are still running today. The not so good ISA BUS or the bus used in most PC Clones is finding it's way into industry more every day. So why would STD bus be better than any of these others.

As I see it, two reasons make it still ideal for industrial control. The first and probably most important is number of vendors supporting the product. The more making the cards, the lower the prices and the greater options available. The ideal situation would be all cards needed available from other vendors (don't have to build any that way).

The second feature to me is size and design. Unlike some other buses, the STD bus is small, compact and designed mostly for simple 8 bit work. Although 32 bit variations are available, the idea is to use 8 bit CPU's and limit the amount of work to within the abilities of the CPU of choice. And yes, almost all CPU's have been built on standard bus cards.

A typical STD system might have one CPU board (CPU, RAM, ROM, Serial and Parallel I/O), and then special digital I/O boards (48, 64, 128, lines of Ins or Outs), or Analog boards (8, 16, 32, or 64 Analog to digital inputs, or digital to analog outputs). Communications is typically RS485 or RS422, which are high speed multiple station serial protocols (one master, multiple slave units.) Synchro and Servo boards, as well as PLC (ladder logic) boards are also available. Now days even PC Clone compatible systems with hard drives and SVGA monitor systems in small 6 inch by 12 inch cabinets are available.

GOOD DEBUGGERS

One difference I have found with standard bus is the typical ROM based debugger. While the newer systems are all DOS based and use LARGE C based development programs that require massive amounts of computing power to run, the typical debugger in ROM allows you to test and often reassemble the whole program. Or simply put, the ROM's are built for people to use to get the program running and tested.

Forth development systems are available, that give you a full Forth in ROM. Whether Forth or BASIC, or just assembly based, these ROM based tools have simple menus, a command line editor to leave code in memory, and test tools to see if it all worked as planed. 8-Bit systems are manageable by normal human beings. You can actually understand and modify these programs without needing 10 or 15 years of hands on experience. This too helps the STD stay viable. When other buses became more complex, simple OEM's had no choice but to stay with STD products or farm out development to other more expensive vendors.

New Options

The newest kid on the block is the PC104 and PCI bus products. Pro-Log who started the STD bus is moving to the PCI industrial version as the new and upcoming replacement for the standard bus. These buses are all based on the PC Clone design and basically are popular because you can migrate your development form the PC to the smaller and more rugged platforms.

The PC-104 is just the ISA bus (PC Clone) put on header pins. This allows a smaller, about 4 inch square product, to be built. The items are stacked one a top each other and a full SVGA, Hard disk, system can be put in a 4 inch cube. The last Embedded conference had these machines sitting on top of demo 17 inch monitors everywhere.

The PCI is the newest bus upgrade to PC Clones and is suppose to be Forth in ROM for intelligent interfacing. The industrial variant has other features which I need to research more before taking a stand on their good or bad features. As reported last issue, I have heard the bus is slower, but has other features that make it more ideal for use. In this case, I think about another two years is needed before we know just how well this product will do.

Homebrew Projects.

For home brewing up an industrial controller, just about anything will do. If the product is really small and needs only a few bits of I/O, you might try one of the newer embedded controllers that I reviewed in issue #72. Old XT's or Kaypros will even provide the bases for using the parallel port for I/O and certainly BASIC will work as a programming language.

If your application needs more complex I/O, or just bunches of it, then STD or S-100 might be a choice. If money is no problem, then use one of the many digital I/O cards for the PC Clone bus. I find those products for the most part over priced, but if you only need one, your time is probably worth more than the overcharging.

Design a board, well not normally. I have just finished several designs based on the 8051 since we were unable to find ones already made (the most common reason for making your own). Our company is also in the business of making I/O products and thus justified in doing original work. And yet the prototype was tested on one of our proprietary bus products. So if it turns out you need to build it, you might consider using one of the buses to test the product before laying your money down on PCB prototypes (expensive to prototype). The CPUZ180 prototype work was done on the S-100 bus.

Tips on Trouble shooting

The standard practice with any bus product is trouble shoot by substitution. That means having a complete set of spares (one for each type) and a spare set of software ROMs if used. The most important tool will be the ROM based debugging tools. I like to use and test known running systems before being tossed out to the job site. That way I understand what the tools actually can provide and what they can't.

One ROM we use can do dumps of memory, 8 bytes at a time. A good aid in trouble shooting is having a listing of RAM from a good running system. To do that use Pro-Com/PCPLus or any good modem program, write a script that asks for all the RAM addresses you want, then upload it to the computer with the send spacing set at max. What happens is every request line of text gets a response, and you turn on the capture feature (saves data to disk) running in half duplex (only see response). This took about an hour, but I was able to get a complete map and data capture of 32K of RAM. A terrific trouble shooting aid which I did use later to solve a problem.

I can't stress enough the need to learn how to use tools of all kinds. I use List (a file viewer) to see binary files, first in text mode to see where the text strings are and then later in hex mode to see code groups. There are many viewers available, just get one and learn all about it. Get a bunch of dis-assemblers for your CPU of interest. I often find bugs from the assemblers or C-Compliers are the source of the problem, and knowing what it actually put out and not what you though it put out can solve many a problem.

For BUS problems, know what makes a minimum system. Start with the CPU only. Does it work? How well? Then add one card and see if it still works. Continue on till problems develop. Write programs in small steps or modules so you can test it in the same way. Load or burn a ROM that only does one thing. When that one thing works, add next step. Only have as small a part of the code as possible to see if it is working, never more. Avoid programming using languages that require it all before you can do a test. Even C can be done is small chunks, although most programmers don't (and thus need big bad debuggers to see what went wrong.)

Lastly think small. Break projects or problems into small parts. See if some task can be done outside the main program. Avoid in line coding and macros, since they often bloat the program for marginal gains and often make trouble shooting almost impossible. Always keep in mind the idea that I will have problems and thus how does the hardware and software design aid me in finding the solution(s).

Signal Descriptions

Power Buses (Pins 1-6 and 53-56). The dual power buses accommodate logic and analog power distribution. As many as five separate power supplies can be used with two separate ground returns as shown in figure 1-3. Pins 5 and 6 provide for alternate use. If used for their alternate purpose these pins shall provide for disconnect capability on the card for conflict resolutions.

Data Bus (Pins 7-14). (8-bit, bidirectional, 3-state, Active-High). Data Bus direction is controlled by the current master and is affected by such signals as read (RD*), write (WR*), and interrupt acknowledge (INTAK*).

All cards should release the data bus to a highimpedance state when not in use. The permanent master shall release the data bus in response to bus request (BUSRQ*) input from a temporary master, as in DMA transfers.

The Data Bus lines may be Multiplexed for address space expansion. The pin assignment for address expansion shall be as shown in figure 1-2.

Address Bus (Pins 15-30). (16-bit, 3.-state, ActiveHigh). The address originates at the current master. The permanent master shall release the address bus in response to a BUSRQ* input from a temporary master.

The address bus provides 16 address lines for decoding by either memory or I/O. Memory request (MEMRQ*) and I/O request (IORQ*) control lines distinguish between the two operations. The particular microprocessor that is used determines the number of address lines and how they are applied.

The address bus may be extended by multiplexing on the data bus. The pin assignment for address expansion shall be as shown in figure 1-2.

Control Bus (Pins 31-52). The control bus signal lines are grouped into five areas: memory and I /0 control, peripheral timing, clock and reset, interrupt and bus control, and serial priority chain.

Memory and I/O Control lines provide the signals for fundamental memory and I/O operations. Simple applications may only require the following six control signals. All STD BUS cards shall support the memory and I/O control lines.

PIN 31 WR*-Write to memory or output (3state, active-low). WR* originates from the current master and indicates that the BUS holds or will hold valid data to be written to the addressed memory or output device. WR* is the signal which writes data to memory or output ports.

PIN 32 RD*-Read from memory or input (3state, active-low). RD* originates from the current master and indicates that it needs to read data from memory or from an input port. The selected input device or memory shall use this signal to gate data onto the BUS.

PIN 33 IORQ*-I/O request (3-state, active-low). IORQ* originates from the current master and indicates an I/O read or write or a special operation. It is used on the I/O cards and is gated with either RD* or WR* to designate I/O operations. For some processors, IORQ* is gated with other processor signals to indicate a special operation, IORQ* with STATUS 1* (MI*) indicates interrupt acknowledge for the Z80.

PIN 34 MEMRQ*-Memory request (3-state. active-low). MEMRQ* originates from the current master and indicates memory read or memory write operations or a special operation. It is used on memory cards and is gated with either RD* or WR* to designate memory operations. For some processors, MEMRQ* is gated with other processor signals to indicate a special operation, MEMRQ* with STATUS 1*(DT/R*) and STATUS 0* (SS0*) indicates Passive for the 8088.

PIN 35 IOEXP-I/O expansion (high expand, low enable). IOEXP may originate from any source and should be used to expand or enable I/O port addressing. An active-low shall enable primary I/O operations. I/O slaves shall decode IOEXP.

PIN 36 MEMEX-Memory expansion (high expand, low enable). MEMEX may originate from any source and should be used to expand or enable memory addressing, An active-low shall enable the primary system memory. MEMEX may be

used to allow memory overlay such as in bootstrap operations. A control card may switch out the primary system memory to make use of an alternate memory. Memory slaves shall decode MEMEX.

Peripheral Timing Control Lines provide control signals that enable the use of the STD BUS with microprocessors that service their own peripheral devices. The STD BUS is intended to service any 8-bit microprocessor. Most peripheral devices work only with the microprocessor they are designed for. Four control lines of the bus are designated for peripheral timing. They are defined specifically for each type of microprocessor, so that it can best serve its own peripheral devices. In this way, the bus is not limited to only one processor.

PIN 37 REFRESH'-(3-state, active-low). REFRESH* may originate from the current master or from a separate control card and should be used to refresh dynamic memory. The nature and timing of the signal may be a function of the memory device or of the processor. In systems without refresh, this signal can be any specialized memory control signal. Systems with static memory may disregard REFRFSH.*

PIN 38 MCSYNC*-Machine cycle sync (3-state, active-low). MCSYNC* shall originate from the current master. This signal should occur once during each machine cycle of the processor. MCSYNC* defines the beginning of the machine cycle. The exact nature and timing of this signal are processor-dependent. MCSYNC* keeps specialized peripheral devices synchronized with the processor's operation. It can also be used for controlling a bus analyzer, which can analyze bus operations cycle-by-cycle.

MCSYNC* should be used to de-multiplex extended addressing on the data bus.

PIN 39 STATUS 1*-Status control line 1 (3state, active-low). STATUS 1* shall originate from the current master to provide secondary timing for peripheral devices. When available, STATUS 1* should be used as a signal for identifying instruction fetch.

PIN 40 STATUS 0*-Status control line 0 (3state, active-low). STATUS 0* shall originate from the current master to provide additional timing for peripheral devices.

Interrupt and bus control lines allow the implementation of such bus control schemes as direct memory access, multiprocessing, single stepping, slow memory, power-fail-restart, and a variety of interrupt methods. Priority for multiple interrupts or bus requests can be supported by either serial or parallel priority schemes.

PROCESSOR	NO. OF MEM ADDRESS LINES	ADDRESS LINES DURING REFRESH	No. of I/O Address Lines	
			I/O MAPPED I/O	MEMORY MAPPED I/O
8080	16	—	Lower 8	16
8085	16	—	Lower 8	16
Z80	16	Lower 7	Lower 8	16
6800	16	—	—	16
6809	16	—	—	16
6502	16	—	—	16
NSC800	16	Lower 8	Lower 8	16
8088	20	—	Lower 16	20

Figure 1-4. Examples of Address Bus Utilization

PROCESSOR	REFRESH* Pin 37	MCSYNC* Pin 38	STATUS 1* Pin 39	STATUS 0* Pin 40
8080	--	SYNC*	M1*	--
8085	--	ALE*	S1*	S0*
NSC800	REFRESH*	ALE*	S1*	S0*
8088	--	ALE*	DT/R*	SS0*
Z80	REFRESH*	(RD+WR+ INTAK)*	M1*	—
6800	—	φ1*	VMA*	R/W*
6809	—	EOUT* (φ2*)	—	R/W*
6809E	—	EOUT* (φ2*)	LIC*	R/W*
6502	—	φ2*	SYNC*	R/W*

*Low-level active
— Not used
R/W* Read high, write low
DT/R* Data transmit high, receive low

Figure 1-5. Peripheral Timing-Control Lines for Various 8-Bit Microprocessors

STD Vendors

These vendors still sell 8 bit CPU based cards. Vendors no longer selling any 8-bitters are not listed.

Computer Dynamics (Z80/6809), 803-627-8800.
Datricon/Scientific Tech. Inc. (6809,68008), 800-221-7060.
Matrix Corporation (Z80/6809), 800-848-2330.
Micro-Aide Corp. (Z80), 818-915-5502.
Micro-Link Tech. Inc. (Z80/8085), 800-428-6156.
Micro/Sys Inc. (Z80), 818-244-4600.
Microcomputer Systems Inc. (8051, NSC800, Z80), 504-769.2154.
Mitchell Electronics (Z80), 614-594-8532.
Robotrol Corporation (Z80), 408-683-2000.
VersLogic Corp. (Z80), 800-824-3163.
WinSystems Inc. (Z80, 64180, 8085 NSC800), 817-274-7553.
XYZ Electronics (6809, 68008), 800-852-6822.
Zwick (64180), 613-726-1377.

TCJ Center Fold

Special Feature

All Users

STD BUS I/O

STD BUS I/O

For this special on the STD BUS or Standard BUS it seems only appropriate to show the BUS as the center fold. To aid in building interface cards to this and other buses, I have also included a sample device address decoding and interfacing diagram.

The actual BUS pins are very similar in functioning to those of S-100. Since more than 8080 type devices have been used on the STD bus, it is a pretty much general purpose hand shake selection of signals. All buses need the DATA path lines and Address signal lines to function. The other signal lines determine when, what direction, and what type of data is being transferred.

Since most Intel derived devices have I/O addressing and do not normally talk to I/O devices using memory addressing, the IOREQ line is used to signal this condition. /MEMREQ is used to signal a memory request in process and would be part of any address decoding scheme. There is only two lines for handling Interrupts, /INTRQ and /NMIRQ, with NMIRQ being the non-maskable interrupt line. All other interrupts where intended to be of the type that put the interrupt address on the data bus when they receive the INTA (acknowledge signal) active. On simple systems, you may decide to just poll devices whenever and interrupt happens and thus us an open collector interrupt line which anyone can pull active.

In laying out the sample design, I chose to use the 138 simply because it is cheap and well understood. It will be found in more old designs than any other device for decoding. To make this design decode 256 byte segments of memory, the upper address lines will need to be decoded. Normally you will put the I/O in one bank of upper memory, say FF00. To decode this you could use another 138, but more typically you would just use a 4 by ONE AND device. Address lines A12, 13, 14, and 15 would feed this device and when all four are HIGH we would have an enable for bank F000. If we only wanted the upper bank, another 4 by ONE AND would be used. To select one of the upper 8 of 16 possible banks, another 138 would be used. This time A8, 9, and 10 would go to A, B, and C of the 138. A11 is hooked to G1 to enable the device only when it is high, thus giving the upper 8 addresses.

Their are many variations and simple modifications you can do to this circuit depending on what extra devices you may have to chose from. I have seen two AND gates used to drive G2A and G2B hooked to A12 to 15 to achieve the same goal. An item to remember is keeping it simple. You want to be able to change the addressing with simple jumpers or functionally by doing a few trace cuts and jumpers. PALs and other changeable devices work well if you have the extra money and equipment to do the changes. The total cost of interface chips is about $3.00, while most PALs will cost over that for just one device and several would be need to provide the same selection of address options.

There has been some new improvements in the STD BUS over the years. Most new products are based on the 386/486 line of CPU's. I consider this over kill for many applications and especially those used when the standard was developed. Today however, entire plants are run from one STD BUS system, and 486s are being put to the maximum use. My feeling is that the bus was not designed for this type of use and you are probably better off changing buses. To this end several variations are possible as discussed in the main article on page 21.

A major problem with the STD bus products seems to be a lack of I/O devices. When checking out recent project options, I noted an absence of high count I/O ports. Typical is 48 IN or OUT as maximum on one card. If you need a thousand points, you quickly run into size problems. This fact is actually true on most other platforms as well. What many are doing to resolve this problem is going to distributed processing. To me the original STD BUS designs were ideal for this condition.

These early designs were simple, limited number of points, and used small CPU's. Those designs are ideal for distributed processing. Limit the number of operations, provide a little buffering, not a large amount of I/O and you have very fast, easy to work on, reliable distributed remote I/O. As vendors find big is not necessarily better, I feel they will start moving back to smaller platforms like the STD bus to solve their large problems. Instead of having one thousand point monsters, they will go to ten locally placed five card STD BUS systems.

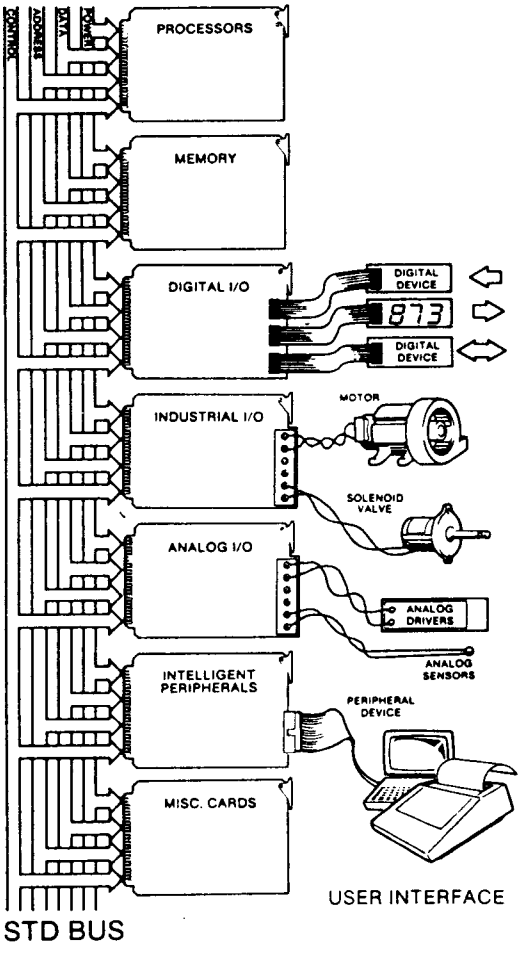
	COMPONENT SIDE				CIRCUIT SIDE			
	PIN	SIGNAL NAME	SIGNAL FLOW	DESCRIPTION	PIN	SIGNAL NAME	SIGNAL FLOW	DESCRIPTION
LOGIC POWER BUS	1	Vcc	In	Logic Power (+5 VDC)	2	Vcc	In	Logic Power (+5 VDC)
	3	GND	In	Logic Ground	4	GND	In	Logic Ground
	5	VBB #1/VBAT	In	Logic Bias #1/Bat Pwr	6	VBB #2/DCPD*	In	Logic Bias #2 Pwr Dwn
DATA BUS	7	D3/A19	In/Out	Data Bus/Address Ext	8	D7/A23	In/Out	Data Bus/Address Ext
	9	D2/A18	In/Out		10	D6/A22	In/Out	
	11	D1/A17	In/Out		12	D5/A21	In/Out	
	13	D0/A16	In/Out		14	D4/A20	In/Out	
ADDRESS BUS	15	A7	Out	Address Bus	16	A15	Out	Address Bus
	17	A6	Out		18	A14	Out	
	19	A5	Out		20	A13	Out	
	21	A4	Out		22	A12	Out	
	23	A3	Out		24	A11	Out	
	25	A2	Out		26	A10	Out	
	27	A1	Out		28	A9	Out	
	29	A0	Out		30	A8	Out	
CONTROL BUS	31	WR*	Out	Write to Memory or I/O	32	RD*	Out	Read Memory or I/O
	33	IORQ*	Out	I/O Address Select	34	MEMRQ*	Out	Memory Address Select
	35	IOEXP	In/Out	I/O Expansion	36	MEMEX	In/Out	Memory Expansion
	37	REFRESH*	Out	Refresh Timing	38	MCSYNC*	Out	CPU Machine Cycle Sync.
	39	STATUS 1*	Out	CPU Status	40	STATUS 0*	Out	CPU Status
	41	BUSAK*	Out	Bus Acknowledge	42	BUSRQ*	In	Bus Request
	43	INTAK*	Out	Interrupt Acknowledge	44	INTRQ*	In	Interrupt Request
	45	WAITRQ*	In	Wait Request	46	NMIRQ*	In	Nonmaskable Interrupt
	47	SYSRESET*	Out	System Reset	48	PBRESET*	In	Pushbutton Reset
	49	CLOCK*	Out	Clock from Processor	50	CNTRL*	In	AUX Timing
	51	PCO	Out	Priority Chain Out	52	PCI	In	Priority Chain In
AUXILIARY POWER BUS	53	AUX GND	In	AUX Ground	54	AUX GND	In	AUX Ground
	55	AUX +V	In	AUX Positive (+12 VDC)	56	AUX -V	In	AUX Negative (-12 VDC)

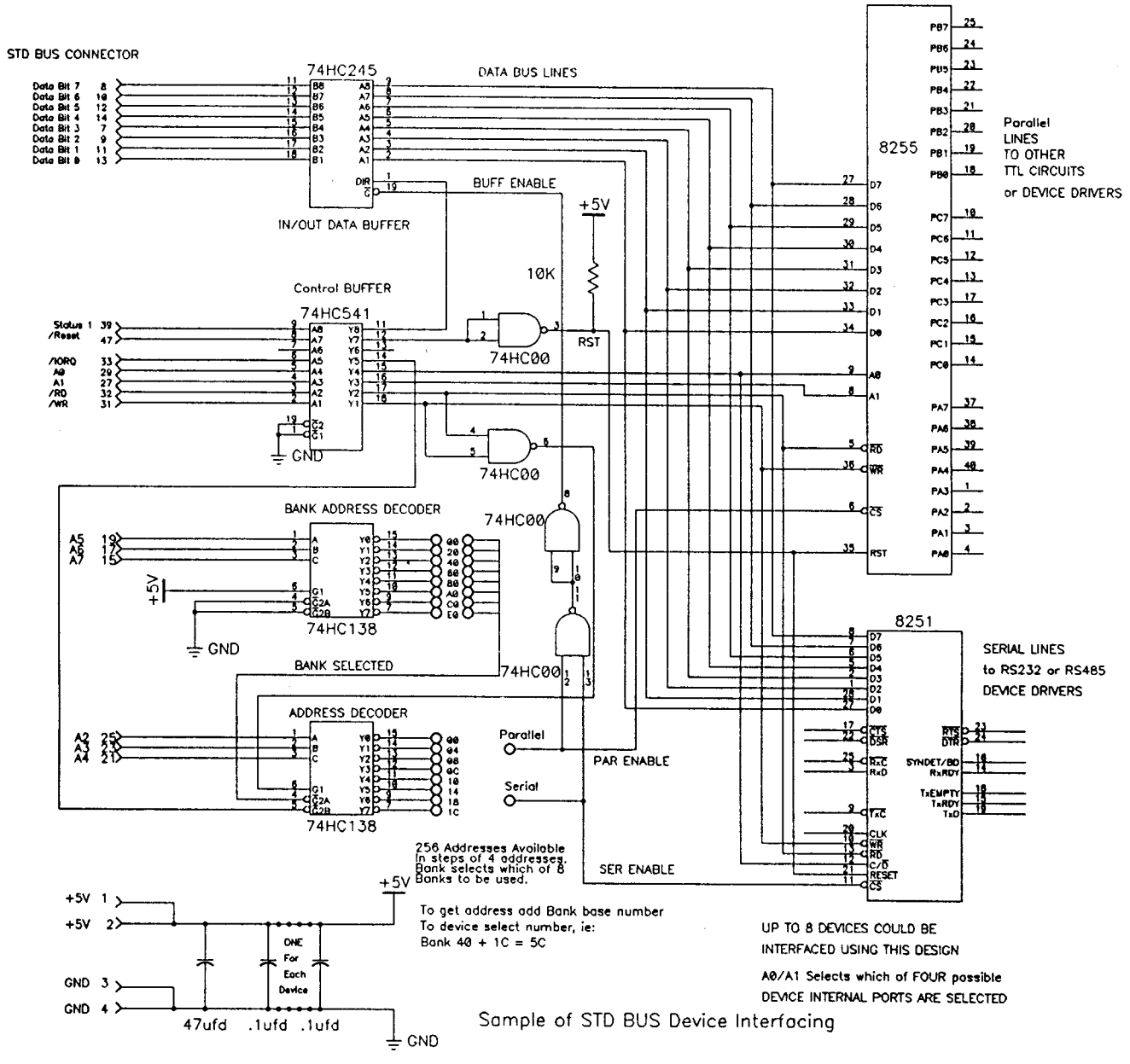
* Low-level active indicator

PIN	DESCRIPTION	COMMENT
1 & 2	Logic Power	Logic Power Source (+5 VDC)
3 & 4	Digital Ground	Logic Power Return Bus
5	Logic Bias Voltage	Low-current Logic Supply #1 (-5 VDC)
*5	Battery Backup Voltage	Alternate use as Battery Backup Voltage
6	Logic Bias Voltage	Low-current Logic Supply #2 (-5 VDC)
*6	DC Power Down	Alternate use as DC Power Down Signal
53 & 54	Auxiliary Ground	Auxiliary Power Return Bus
55	Auxiliary Positive	Positive DC Supply (+12 VDC)
56	Auxiliary Negative	Negative DC Supply (-12 VDC)

*PIN 5 VBAT—Battery Backup Voltage. VBAT is a DC voltage.

PIN 6—DCPD DC Power Down Signal. DCPD* is a logic signal that indicates Vcc has dropped below the recommended operating limit.





Sample of STD BUS Device Interfacing

PARAMETER	LIMIT	REFERENCE
Positive voltage applied to logic input or disabled 3-state output	+Vcc + 0.5 V	GND pins 3,4
Negative DC voltage applied to a TTL logic input or disabled 3-state output	-0.4V	
Negative DC voltage applied to a CMOS logic input or disabled 3-state output	-0.5V	

Maximum Voltage Ratings

CARD PIN	SIGNAL NAME	SUPPLY VOLTAGE	TOLERANCE	REFERENCE
1,2	TTL Vcc	+5V	±0.25V	GND pins 3,4
1,2	CMOS Vcc	+5V	±0.50V	GND pins 3,4
5	VBB #1	-5V	±0.25V	GND pins 3,4
5	VBAT	*	—	GND pins 3,4
6	VBB #2	-5V	±0.25V	GND pins 3,4
55	AUX +V	+12V	±0.5V	AUX GND pins 53, 54
55	AUX -V	-12V	±0.5V	AUX GND pins 53, 54

*VBAT may range from +3.5V to Vcc

Power Bus Voltage Ratings

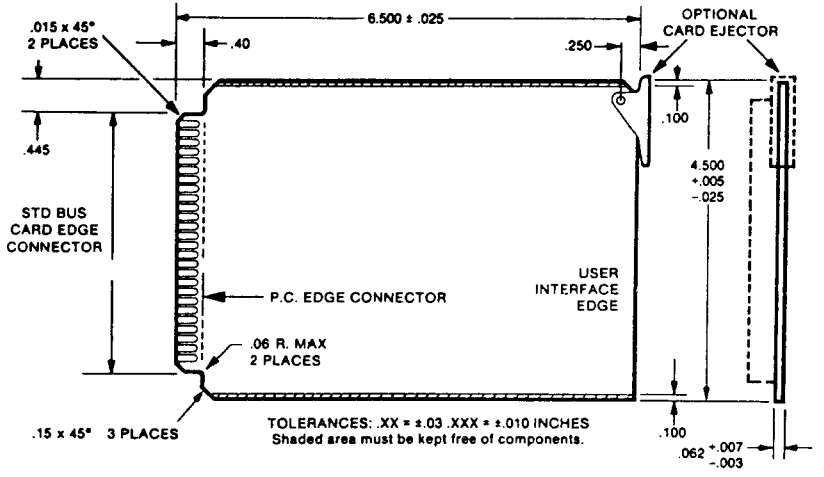


Figure 1-17. Bus Card Outline—Inches

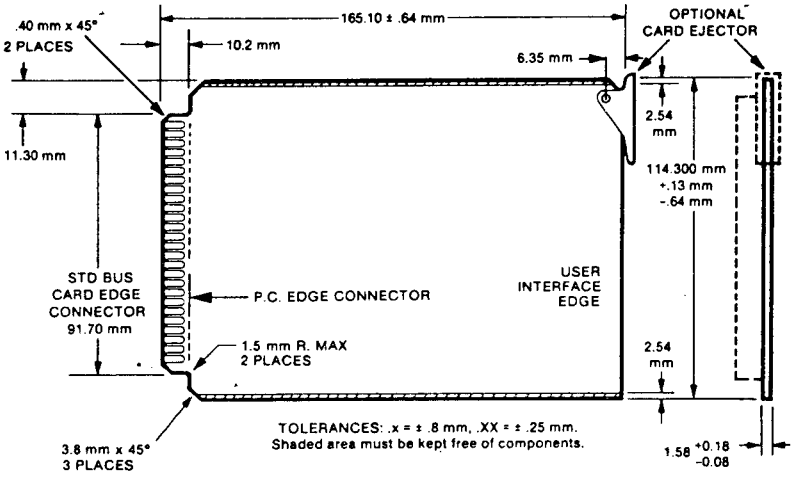
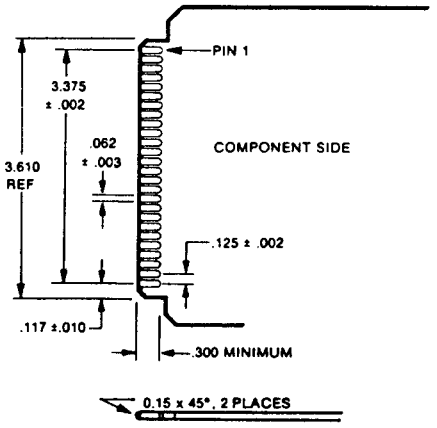
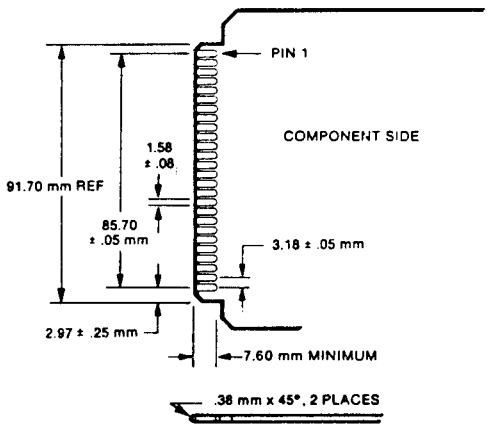


Figure 1-19. Bus Card Outline—Metric



Bus Edge Card Finger Design—Inches



Bus Edge Card Finger Design—Metric

EPROM Simulator

By Terry Hazen

Special Feature

Classic Support

Hardware Help

A Simple EPROM Simulator for CP/M Systems

I wanted to upgrade the boot EPROMs in my 8mhz Ampro Z80 and 18mhz Yasbec Z180. I've been working on system software to make both systems faster and more responsive. One thing I really wanted was a way to speed up the cold boot process. I wanted to try booting an entire banked operating system directly from EPROM. That would give me an "instant" cold boot by eliminating time-consuming disk reads. Since both systems have RAM disks, I could make them the system disk and substitute the fast RAM disk initial login for a much slower floppy or hard disk login.

The EPROM development process tends to become a repetitive and time-comsuming cycle of writing test EPROM code, blowing a test EPROM, plugging it in and testing it, removing it and replacing it with the standard boot EPROM to reboot the system, revising the code, blowing a new one, installing and testing it, etc, until things are finally working right.

I decided the best way to improve the process was to make a simple EPROM simulator by substituting some Static Ram (SRAM) for the EPROM. Since CP/M systems perform their cold boot under EPROM control, they allow direct access to the boot EPROM under program control. For example, my Ampro and Yasbec can both map the EPROM into the lower 32k of the Transient Program Area (TPA) when a program sends a control byte to one of the system ports.

With an EPROM simulator, I could develop code, copy it into the substitute SRAM under program control and test it by rebooting from the SRAM. The SRAM should be battery-backed and it would also be nice to be able to switch the original boot EPROM back into the circuit so I could easily reboot in case (or when) my latest code revision failed.

Giving credit where credit is due, Ludo Vanhemelryck got me started in this direction last year when he described an auxillary SRAM/EPROM board he'd built for his Yasbec. He'd used one of the Dallas Semiconductor Nonvolatile SRAM modules that combine a SRAM and its battery backup in one convenient epoxy module with a pin-out almost identical to the EPROM pinout. Since I wanted to simulate the 32x8 27C256 256k EPROM, the DS1230Y-200 256k Nonvolatile SRAM looked ideal. The DS1230Y is readily available directly from Dallas Semiconductor as well as from Jameco and JDR Microdevices.

Based on Ludo's board and my own needs, I came up with a simple EPROM simulator for my Yasbec consisting of a 2.00" x 2.50" single-sided PCB that plugs into the Yasbec EPROM socket. The PCB contains two 28-pin DIP sockets for the SRAM and EPROM, one 28-pin DIP socket that is reversed to serve as a DIP plug, three pull-up resistors and two jumper headers that serve as connectors for two external EPROM/SRAM control switches, and one off-board SRAM control wire.

The Circuit

The EPROM simulator circuit is shown in Figure 1. While this circuit was designed specifically for use with a Yasbec, it should be easily adaptable to almost any 8-bit SBC. Only the size of the SRAM/EPROM, the PCB layout and the method of obtaining the /WE signal will depend on the specific host computer.

Both power pins and most address and data lines from the DS1230Y, (U1) and the 27C256 (U2) are just bussed together and connected to the same pins on the 28-pin DIP plug (P1.) One difference is that the DS1230Y's A14 is on pin 1 and is connected to the A14 pins on 27C256 and P1, which are both on pin 27.

Both the DS1230Y and the 27C256 have Chip Enable (/CE) on pin 20. Each of these pins is connected to a 10k pull-up resistor and to the SRAM/EPROM Select switch (S1.) The switch common is connected to P1 pin 20, allowing switch S1 to select between the SRAM and the EPROM. Pin 1 (Vpp) on both P1 and U2 are left unconnected.

SRAM pin 27 is the Write Enable (/WE) line, which is used to control SRAM writes via the SRAM /Read-Write switch (S2.) The /WE signal must be high for SRAM reads and low for SRAM writes. Since there is no /WE signal directly available at the Yasbec EPROM socket to control the SRAM Read/Write mode, you have to provide it some other way. Luckily the Yasbec has /WE available on a nearby SRAM pin, so I ran a tiny flexible insulated wire from the S2 Read/Write pin and soldered a flat IC pin to the free end as a connector. After I plug the simulator board into the Yasbec SmartWatch

socket, I make the /WE connection by slipping the IC pin into the Yasbec SRAM DIP socket alongside /WE pin 29, being careful that it doesn't short to either of the ajacent pins.

Now, when S2 is in the Read/Write position, the computer controls /WE and automatically switches the SRAM between the Read and Write modes. When S2 is in the Read-Only position, the /WE line is connected to a 10k pull-up resistor to force a Read-Only condition that protects the SRAM from accidental writes.

If you don't have access to an appropriate /WE signal, you can use the brute-force method and connect the S2 Write pin directly to ground. You must now manually use S2 to switch the SRAM between the Read and Write modes. This means that a utility designed to load code to the SRAM with S2 in the Write position can't read it back to verify it until you switch S2 to the Read position.

A more complicated approach is to use a little extra discrete logic or a PAL or GAL to generate an appropriate /WE signal. On the Yasbec, for example, the SRAM /WE signal is produced by a 16V8 GAL:

/WE = /WR * /MEMREQ

The Printed Circuit Board

My Yasbec SBC is located in one slot of a 4-slot card cage. To minimize the height added by the EPROM simulator PCB to keep it within the adjacent card space, I designed it to take maximum advantage of the space available around my Yasbec EPROM socket. The simulator PCB plugs into the socket on top of a Dallas Semiconductor DS1216E SmartWatch module which is plugged into the Yasbec EPROM socket. The simulator EPROM and SRAM sockets are located on the same side of the simulator PCB as the DIP plug, allowing them to hang down on either side of the SmartWatch module.

I laid out the single-sided PCB, punched and drilled the holes, applied Datak dry transfer pads and printed circuit layout tape using the direct-resist method, then etched and tin-plated the board (see TCJ#62 p31 for more details of this method). I couldn't find an appropriate 28-pin DIP plug, so I made one from a standard 28-pin solder-tail DIP socket. I soldered a short length of bus wire into each socket pin position, then slipped the free wire ends through the PCB holes and soldered them. With the socket side against the PCB, the socket solder-tails become the plug pins. Since the DIP plug is mounted on the component sid of the PCB, the actual DIP plug pin numbers on the PCB are reversed side-to-side.

A more general simulator PCB layout approach would be to use a standard 28-pin DIP socket on the simulator board as P1. You can then use a 28-conductor ribbon DIP cable to connect the board to the host computer EPROM socket, allowing you more flexibility in where you mount the simulator PCB.

Switches S1 and S2 are wired to 3-pin header plugs that plug onto the 3-pin switch headers on the simulator PCB. So the switches could be readily visible and easily controlled, I mounted them centered in the narrow face of a 4" piece of 1/2" x 1" aluminum L-extrusion from an old shower door surround frame and slipped the extrusion into a slot in the Yasbec card cage adjacent to the Yasbec board.

Loader Utility

In order to use the EPROM simulator, I needed to write a simple loader utility to remap the EPROM/SRAM into the lower 32k of the system address space, which is where both the Ampro and Yasbec access their EPROMs, and to load the code from the EPROM binary code file to the SRAM.

While the actual loader code is completely computer-specific, there are some general principles to keep in mind involving memory remapping. It's easy to forget that the loader program can't occupy the space that's being remapped! If the EPROM or SRAM is being remapped into the lower 32k, the loader code can't run in low memory at the usual 100h. Instead, it must run above 8000h. Also, since the usual BIOS and BDOS access points and the IOBYTE in page 0 of lower memory aren't available when the SRAM is remapped into that address space, you can't use BIOS or BDOS calls while the SRAM is occupying lower memory.

In ZCPR33 or ZCPR34 systems, the loader can be a Type 3 utility, which loads and runs at a specified address, typically 8000h. In the less-developed standard CP/M world, the loader program code must include extra code to relocate the working part of the loader code from low memory where it is initially loaded, to high memory above the bankswitching area before it is run. There are several ways to do this. Bridger Mitchell's Advanced CP/M column in TCJ#33 (you have ordered all those TCJ back issues, haven't you?) has an elegant example of Z80 relocation code.

With the loader code located out of harm's way, you can use the standard CP/M BDOS Read_Sequential and Set_DMA functions to read the file records in sequence into a buffer in high memory. Then you can remap the EPROM/SRAM into the lower 32k of the system address space. For the Ampro, this is simply done by sending a control byte to the Ampro system control port (port 0.) In Z80 code, the routine is:

```
LD      A,0     ; Select EPROM
OUT     (0),A
```

With the SRAM accessible and S2 set to Read/Write, you can copy the code from the buffer to the SRAM address space starting at 0000h. Then you should compare the contents of the SRAM with the buffer to make sure the SRAM is correctly

PADS
by Terry Hazen

When I was working on my SCSI EPROM programmer (TCJ#62,63), contributing editor Tim McDonough sent me a copy of PADS, a DOS shareware PCB and schematic CAD utility. At the time, I didn't have a PC to run it on and I had to rely on our kind editor to convert my hand-drawn schematic to CAD format for the article. Since then I've picked up an inexpensive used 386 notebook PC, partly so I could try out PADS on my EPROM simulator project. I'd already made my PC board by the time I got the notebook, so this PADs evaluation is limited to my experience using PADS-Logic to generate and laser-print the EPROM simulator schematic.

The shareware version of PADS is a somewhat limited version of the commercial PADS packages by PADS Software. No registration is required for the evaluation package. PADS Software gets its money if you choose to upgrade to their commercial packages. It came on 3 disks, PADS-Logic, PADS-PCB and a library disk.

I'm not sure what all the limits of the PADS evaluation package are. The documentation for PADS-Logic says that it's limited only in the size of the circuit you can create. Tim said that it's limited to designs with less than 14 ICs, which is good enough for most of us. The documentation for PADS-PCB says that it's limited to about 30 IC's.

According to the documentation:

"With the PADS-Logic shareware package, you can create schematics, make plots, produce net lists, define library parts... all of the functions of PADS-Logic. With the PADS-PCB shareware package, you can automatically place your design, autoroute it, create photoplot outputs and N/C drill files... all of the functions of PADS-PC. All drawings, library parts, and layouts you create can be read directly by the commercial versions of PADS-Logic and PADS-PCB."

PADS is a DOS program that requires the usual stuff - a 286 or better, DOS 3.3 or better, EGA, VGA or equivalent graphics and a mouse, plus 7 megabytes of available hard disk space. Since it displays things in various colors, it works best with a color monitor, but I was able to use it with my monochrome LCD notebook display with few problems.

The PADS shareware documentation comes on disk and should be printed out before you attempt to install PADS. Print out the README.DOC file on any of the disks for more detailed information. While the printed documentation makes an impressive pile, it's not very useful when you are trying to find specific information on how to do something. In fact, it's not very easy to find things at all, as there is no index in the manual and no on-line help available. You really need to put index tabs in the printed manual to help you find pertinent sections.

Learning to use PADS is like learning to use any CAD program - a slow and frustrating process. But once you begin to get the hang of it through (seemingly endless) experimentation, you can produce a schematic fairly readily. I found that saving partial schematics under a series of different filenames during the learning process gave me the freedom to experiment, goof up and still be able to go back to the last good version for another try. The schematic files for the EPROM simulator only took up about 24k, so I saved lots of versions.

In the process of generating my schematic, I used several library component patterns and because the available patterns didn't always fit my needs, I had to create or modify several others and add them to the library. The process for working with library patterns isn't covered very well in the documentation but you can figure it out if you persist and keep experimenting.

PADS allows you to zoom in on details for ease in placing components, connections or text. However, text seems to obey different zoom rules than the rest of the schematic, with disconcerting and confusing results. If, for example, you place a word in the center of a box and then zoom in on it, the text might appear larger and longer and overflow the box. If you zoom in further, it might again be centered. That makes it difficult to select the proper text size and placement. Because of the display inconsistencies, it's hard to tell exactly how text is going to print out in relation to the rest of the schematic until you try it.

PADS can plot schematics on dot-matrix or laser printers as well as plotters, etc, and it can generate (large!) files with included printer control codes that can be sent to a printer later. It can also produce ASCII files that it says can be read by other schematic CAD programs. The finer details of schematics printed on my old Epson MX-80 dot-matrix printer weren't always very readable, but the plots were more than good enough for checking output formatting. The EPROM Simulator schematic was printed by PADS on a friend's laser printer.

The PADS-Logic evaluation package worked well enough that I didn't run into too many limits in my simple project. It's certainly a useful enough tool to spend the time required to learn it, and the price is right. Whether I'll ever be motivated to upgrade to the full packages or not is another matter entirely. The weakest part for me was the PADS documentation. If the PADS evaluation package is intended to entice me to buy the commercial PADS packages, I would expect the manual to be at least as good or even better than the commercial manual. But it isn't even barely adequate, especially compared to the manuals I've seen for commercial DOS and Windows CAD packages. My learning time would have been radically reduced if I'd had a better manual that included complete command references and a comprehensive index.

loaded, which helps if you forgot to set the SRAM to Read/Write or if you have the EPROM selected instead of the SRAM.

Finally, restore the original lower system bank. The Ampro routine is:

```
LD      A,41h   ; Reselect system RAM
OUT     (0),A
```

Now you can clean up and exit back to the system. That's all there is to it.

Results

This simple EPROM simulator was invaluable in helping me develop new boot EPROMs for my Ampro and Yasbec and is a great addition to my computer hardware tool set for future work on dedicated controllers. How did the "instant" cold boot work out? After the initial power-on cold boot, which still takes quite a bit of time because both my terminal and SCSI hard disks do extensive power-on self-testing, the Ampro now takes about 3 seconds for a complete cold boot, including running two setup utilities from the hard disk during the boot process. The Yasbec is a bit faster and can do the same thing in about 2.5 seconds. Eat your heart out, Windows users!

References

1) Dallas Semiconductor (1-800-336-6933)
2) Jameco (1-800-831-4242)
3) JDR Microdevices (1-800-538-5000)

ZCPR33 Type 3 Header

ZCPR3 utilities start with a special ZCPR3 header (see Jay Sage's column in TCJ#55). The simple version of the ZCPR3 header for Type 3 utilities is:

```
;
ENTRY: JP   START    ; Jump past the header
       DB  'Z3ENV'   ; ZCPR3 ID
       DB  3         ; Type-3 environment ID
ENVPTR: DW   ; Address of ZCPR3 ENV module,
             ;filled in by ZCPR33+
       DW   ENTRY    ; Intended load address
START:               ; Remainder of program code
```

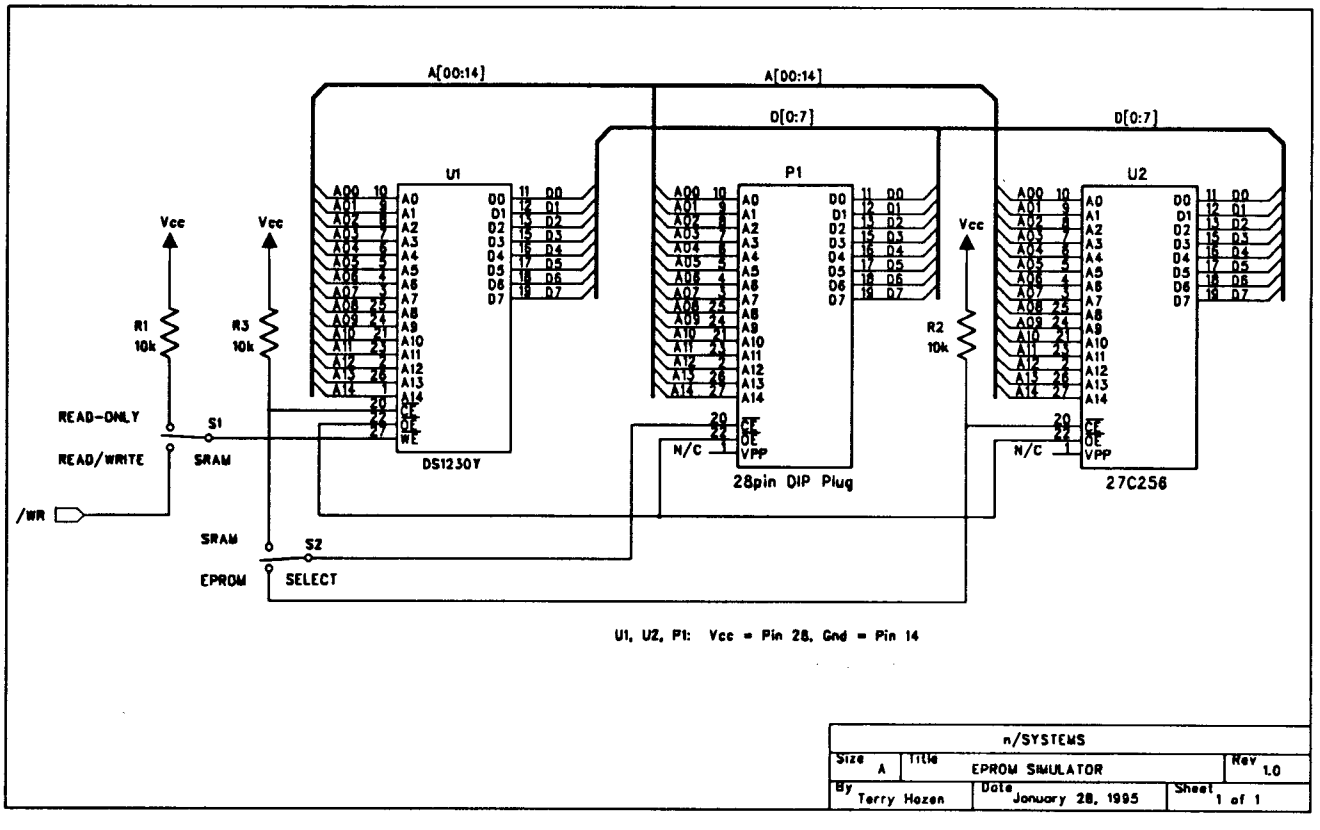
After writing the program code, you create the Type 3 utility by first assembling the source code to a REL file. If you are using Z80ASM or ZMAC to assemble FILENAME.Z80, the command lines to produce FILENAME.REL are:

```
Z80ASM filename/m
ZMAC filename
```

Then the REL file is linked to run at 8000h to produce the file FILENAME.BIN. The SLRNKP and ZML command lines are:

```
SLRNKP filename/n,/a:8000/j,filename,/e
ZML filename /a:8000
```

Finally, rename FILENAME.BIN to FILENAME.COM. Because of the information contained in the ZCPR3 Type 3 header, ZCPR33 now knows to automatically load and run FILENAME.COM at 8000h.



Disk I/O in Forth

By Walter J. Rottenkolber

Special Feature

Beginning Forth

Part 2: F83 Forth

This is part 2 on some fundamental of Forth. Why Forth? The Forth programs available for all the old machines work quite well and can provide the user with advanced tools. The source code is almost always included, so bug fixes and changes are possible and easy once Forth is mastered. Mastering Forth can be a problem due to a lack of books for the beginner. To help those starting down the mastering path of Forth, Walter has undertaken the project of explaining how file I/O is done in FigForth and F83. This is the F83 portion, with FigForth having appeared in issue #74.. BDK.

Laxen and Perry F83 —

Laxen and Perry designed F83, their 1983 Standard Forth, to work with CP/M and MSDOS disk files. These files hold blocks. Once a file is Opened, it behaves as though it were a disk in a pure Forth system.

F83 demonstrates the least used method for paging out a buffer. The block requested least is the one swapped out when the buffers are full.

Unlike figForth, F83 separates the buffers from the control data. The block buffers exist as a group of 1024 byte spaces in RAM. The constant FIRST points to the base of the buffer group, and LIMIT to the top. Located just below FIRST, is an array that holds the block control records (BCR), one for each buffer.

The beginning address of the array is returned by >BUFFERS, its end address by >END, and its size in bytes by >SIZE. BUFFER# (n — addr) returns the address of the nth BCR. BCRs are numbered 1..n, because BUFFER# 0 is used as a temporary record holder. #BUFFERS returns the number of buffers.

Each record is composed of four integers (8 bytes) holding, in order, the Block Number, a Pointer to the CP/M File Control Block (FCB), the Buffer base address, and the Update flag. On Initialization, blocks are numbered -1 to mark them unassigned, the FCB pointer is nulled, the proper buffer address inserted in order, and the update flag reset to zero.

Because the block number is reset to -1, access to block0 is hassle free. Block0 still cannot be loaded, but it provides a convenient title screen to describe the purpose of the file.

The update flag is set when it contains a negative number, usually TRUE (-1). It has two resets. A positive number, usually 1, means the buffer is assigned, but the data is either not read in yet or should be reread. A zero means that the data has been read into the assigned buffer.

The array behaves like a stack with the current BCR on top, and the least used BCR at the bottom. When a block is requested, the Word LATEST? checks the top BCR first. If it isn't the desired block, the array is searched by ABSENT?. LATEST? and ABSENT? check both the block# and the block-FCB before considering it a match. This distinguishes between blocks having the same number but located in different block files. If the block is already assigned to a buffer, its BCR is moved to the top of the array, and the other BCR's pushed down. In this way, frequently requested blocks are more apt to be held in memory.

If it is not present, MISSING assigns the bottom, least used, BCR buffer to it. But first, it checks the Update Flag, and if set, writes out the block. The BCR is then moved to the top of the array making it current. The buffer base address is returned on the data stack.

The basic block Words for a F83 system:

BLOCK	IN-BLOCK	BUFFER
FILE-READ	FILE-WRITE	UPDATE
DISCARD	FLUSH	SAVE-BUFFERS
EMPTY-BUFFERS		

BLOCK (blk# — buff-adr) Returns the buffer address containing the data in block# from the file FCB pointed to by FILE. Reads in the block if not previously done, or if the block was DISCARDed.

IN-BLOCK (blk# — buff-adr) Same as BLOCK except the block is from the file pointed to by IN-FILE.

BUFFER (blk# — buff-adr) Returns the address of the buffer assigned to block# and the file FCB pointed to by FILE. Unlike figForth, it will return the buffer address of a previously assigned block.

Note: In Forth-83, the value in OFFSET is added to the Block# in all three of the above Words. As a result, block counts begin

from the same Block0. Normally OFFSET is zero, but it allows for change if F83 is modified to access disks as a pure Forth system. IN-BLOCK permits reading data in from one file and writing it out to another file via BLOCK or BUFFER.

FILE-READ (BCR) Reads block# from file pointed to by the block control record to assigned buffer.

FILE-WRITE (BCR) Writes buffer block# to file pointed to by the block control record.

Note: Both these Words are vectored through the deferred Words READ-BLOCK and WRITE-BLOCK, respectively. The source code in DIRECT.BLK, which comes with F83, sets up a pure Forth disk system that uses BIOS routines. This could be implemented and re-vectored to allow transfer of figForth data and source to F83 files provided the disk formats are compatible.

UPDATE — Sets the update flag of the current block to force a data write before a block discard.

DISCARD — Resets the update flag of the current block with 1 so that a repeat block access forces a read from disk first (unlike figForth).

SAVE-BUFFERS — Writes out buffers with a set update flag, but leaves the data unchanged. Resets update flag with zero so that reaccess of a block already in a buffer does not cause a reread, but simply returns the buffer address.

EMPTY-BUFFERS — Erase the disk buffers, and reinitialize the block control array. This dumps all buffer data and forces new blocks to read in data.

FLUSH — Combines the function of Save-buffers and Empty-buffers. Use at end of work session or when leaving Forth.

The Kaypro F83 FLUSH has a problem similar to that of figForth. It places the command, 0 BLOCK DROP, between the Save-buffers and Empty-buffers. But this assumes you have not previously read in Block #0. If you have, it will not work, because unless enough non-zero blocks are read in to force a discard of Block #0, you will get just a return of the buffer address, not a disk read. The following definition for the F83 FLUSH adds an extra EMPTY-BUFFERS to ensure that Block #0 will be read in.

```
: FLUSH   SAVE-BUFFERS  EMPTY-BUFFERS
          0 BLOCK DROP  EMPTY-BUFFERS ;
```

F83 also has Words to manage the files that contain the blocks. These are not part of the Forth-83 standard, so other Forths may manage files quite differently.

CREATE-FILE	MORE	OPEN
FROM	CAPACITY	FILE?
SWITCH	DEFINE	SELECT
DRIVE?	RESET	DIR
SAVE-SYSTEM		

Note: I use (S xxx) to comment on string data used by Words. In Forth (and F83), Words use number data with Postfix notation, while string data is Prefix.

CREATE-FILE (#blks) (S d:filename.ext)

Use as: #blks CREATE-FILE d:foofile.blk

This creates a file of 'filename', #blks number of blocks in size. The drive 'd:' is optional. It closes the file afterwards, so to use the file, you will have to OPEN it first.

MORE (#blks) Increases the size of the current file by #blks, then closes file. Enlarged file must be OPENed to use. There is no way to shrink a block file.

OPEN (S d:filename.ext) Searches dictionary for the FCB of the file, creating one if necessary. It then opens the file if it has been previously created, and makes it the current file. Both the FILE and IN-FILE pointers are set to the File Control Block (FCB) of the current file. Prints error message if file is not found.

FROM (S d:filename.ext) Searches dictionary for the FCB of the file, creating one if necessary. It then opens the file if it has been previously created and assigns it to the IN-FILE pointer only. Prints error message if file is not found. Used in concert with OPEN.

Note: Block transfer Words are designed to read from the IN-FILE FCB and write to the FILE FCB. An OPEN file will transfer blocks to itself. When a FROM file is opened, the blocks will now be transferred from the FROM file to the OPEN (current) file. At the end of the transfer, IN-FILE is set the same as FILE by !FILES. This avoids accidents, but also means that another transfer requires the From file to be opened with FROM again.

CAPACITY (— n) Returns the size, in blocks, of the current file.

FILE? Prints the name of the current file.

SWITCH Exchanges the FCB's of FILE and IN-FILE to allow a reverse transfer. You have an Open file and wish to copy TO it from another block file. Do a FROM filename, then SWITCH.

DEFINE (S d:filename.ext) Creates an FCB for the file in the dictionary, but does not open it.

Note: Except for Words that recycle the kernel FCB1 and FCB2, new FCBs are created as dictionary entries. These could be lost during a Forget, and would require a repeat Opening of the file. To protect the FCBs you intend to use, first DEFINE them, and then move FENCE with HERE FENCE !.

SELECT (drv#) Sets default drive (0=A, 1=B, ...). Usually redefined as, : A: [DOS] 0 SELECT ; , etc.

DRIVE? — Prints current drive, e.g. A:.

RESET — Does a Drive Reset on the current drive. Used when changing disks to avoid a BDOS error. You may have to switch in the DOS vocabulary, i.e DOS RESET, to run it. DIR — Like the CP/M version, it types out an unsorted list of all files on the current drive.

SAVE-SYSTEM (S d:filename.COM) Saves the core image of F83 to file along with any additions to the dictionary.

The file Words in F83 are quite simplistic. Normally only one file is open at a time, and it behaves as though it were a disk in a pure Forth system. A From file can be opened temporarily as a read only file to allow for transfers to the current file.

It will also allow you to load screens from another file by having the following line in the load screen:

FROM (S d:filename.ext) n LOAD

LOAD resets the In-file to the current file. This concept can be extended as in INCLUDE and INCL-THRU (screen 1). INCL-THRU works the same as THRU except the screens are from another file. The extra code is needed so you return to the file holding the main load screen.

More modern Forths set up an FCB pointer array and use an index to the array as part of the Open routine. The VIEW Words in F83 use such an array to access source screens, and provides a good example of how it is done.

Forth Blocks

Dealing with Forth blocks takes an attitude readjustment. It helps to be familiar with databases, records, and random access, and to think of data as a discrete unit. One problem facing Forthwrights is that public domain Forths, like other PD programs, have no consistent support. You must be willing to learn new ideas and to experiment with them.

You can try this exercise:

10 CREATE-FILE TEST <cr>
OPEN TEST <cr>
2 BLOCK 60 2DUP BLANK EXPECT <cr>
Then type in "Hello Forth" <cr>.
UPDATE FLUSH <cr>
2 BLOCK 60 TYPE <cr>

Note: <cr> means press the Return key.

This creates and opens the file TEST. EXPECT will wait for you to enter up to 60 printable characters. A <cr> ends the entry of fewer chars. UPDATE causes FLUSH to save the block. The following line reads the block and types out your message. In figForth, you will not need to type the first two lines, but you should check first with LIST that the block is empty.

Working with entire Forth blocks is straightforward. A simple copy routine:

```
: COPY  ( from-blk# to-blk#)
\ SWAP BLOCK SWAP OFFSET @ + BUFFER   \ figForth
        SWAP IN-BLOCK SWAP BUFFER
        FILE @ [ DOS ] !FILES          \ F83
        B/BUF MOVE UPDATE FLUSH ;
```

The Swaps arrange for the from-blk to be read in first, and then the to-blk-buffer to be assigned. The block addresses are adjusted so the stack is (from-adr to-adr 1024) for Move to shift the data from one buffer to the other. Also, requesting Buffer after Block makes it Current so Update sets the proper flag for Flush.

In figForth, block includes Offset, but Buffer does not, so it must be added. For F83, IN-BLOCK allows for a transfer between files, and then FILE @ !FILES resets the system to the current file.

It's possible to renumber the block and write it out. MOVE-SCR in the Kaypro figForth does so, but it is implementation specific.

```
: MOVE-SCR  ( from# to#)  SWAP BLOCK 2 - !
                    UPDATE FLUSH ;
```

To demonstrate, in F83 code, how you can build on a basic copy Word, look at source screens #3..9, which develop routines to copy, clear, shift, insert, delete, and settle screens.

COPY is abstracted into two parts to allow for recycling of code.

COPIES has two subWords, <COPIES AND COPIES>, because a destination between a group of screens would result in overlap. This is prevented by copying from the top down, rather than bottom up, when copying to a Dest# greater than From#. An alternate version does this only if Dest# is between From# and To#. Deliberately overlapping a copy is one method of repeating a group of screens or other data. For example, 3 24 6 COPIES> would repeatedly copy screens #3 #4 and #5 into screens #6..#24.
A keypress aborts a block copies.

The code, 1 1 D+ (n n' — n+1 n'+1), does the same as 1+ SWAP 1+ SWAP, that is, increments the top two integers on the stack. The code 1 1 D- decrements them.

There are two versions of (COPY), <COPIES and COPIES>. The ones commented out move the data from one block to the other. In the more complex version, ESTABLISH is F83's way

to make the move by simply changing the block# and FCB of the current block to that of the new block. The Copies also use a double loop to transfer blocks in batches. In the simple version, a Write would occur for every Read once the buffers were full.

Note: F83 already has a good Copies routine called CONVEY, used as: from# to# TO dest# CONVEY.

SHFT-BLK is a copy that uses an offset to the From# to mark the destination.

ADDSCR and DELSCR add and remove screens in a block file, and blank out duplicate screens. It would have to be extended to work properly with files holding shadow screens (ref 1).

Note: ?QUITBLK uses R> DROP EXIT to cause an exit from the calling Word, ie. ADDSCR and DELSCR.

SETTLE resembles a similar Word in Frank Sergeant's Pygmy Forth. It packs full screens to the bottom of the block file.

You can request a block and save its address momentarily, but it is dangerous to go on to other blocks with the intention of using the address later to access the buffer. The block paging is automatic, so you have no guarantee that the buffer will hold the data you expect when you return. This is especially true of multiuser, multitasking systems. Good practice is to move the such data to previously allocated RAM memory, and to write it out after completing the task.

```
: BLK>MEM  ( mem-base-adr start-blk# #blks)
        BOUNDS  ?DO
        I BLOCK OVER B/BUF MOVE B/BUF +
        LOOP DROP ;
```

This moves #blks beginning at start-blk# to a memory location pointed to by mem-base-adr. And MEM>BLK reverses the process.

```
: MEM>BLK  ( mem-base-adr start-blk# #blks)
        BOUNDS  ?DO
        \ DUP I OFFSET @ + BUFFER    \ fig
        DUP I BUFFER                 \ F83
        B/BUF MOVE UPDATE B/BUF +
        LOOP DROP FLUSH ;
```

Note: : BOUNDS (addr len) OVER + SWAP ;

In these Words you may have noticed certain stack conventions used in Forth programming. Movement of a datum to a variable by (value address). Blocks of data are defined by (address count). Filling a block of data (address count value). Moving a block of data (from-adr dest-adr count). Using addresses so openly gives Forth an assembler like quality not found in Basic or C.

You may not see the Block Words because they are often buried in the Words that use them. LIST prints out the Forth source screen. These contain only printable characters from Blank to Tilde (~). No control or hi-bit bytes. The Forth interpreter wants a stream of bytes containing only Words and number strings separated by blanks. When a screen is printed as 16 lines of 64 characters, it's an artifact of the screen list or editor program.

```
: LIST  ( #blk)
        BLOCK  16  0                    DO
        CR  I 3 .R SPACE  DUP C/L TYPE  C/L + LOOP
        DROP CR ;
```

The CR forces a newline. The constant C/L, returns the number of characters per line (64). Block places the buffer address on the stack. After printing the line number, C/L characters are typed out, the address increased by C/L, and the loop repeated until all 16 lines are printed. F83 would expand this with the IN-BLOCK !FILES duo as in COPY so you could List a block from another file without losing the current file.

It's possible to locate and access only a portion of a block. FigForth uses the Word .LINE to print out a single line from a block of text. When used for system/error messages it saved valuable RAM in early systems.

```
: (LINE)  ( ln# start-blk# — addr len)
   >R C/L B/BUF */MOD R> B/SCR * + BLOCK + C/L ;

: .LINE  ( ln# start-blk#)  (LINE) -TRAILING TYPE ;
```

The calculation is equivalent to:

```
LN# * C/L  —>  Total offset to line
B/BUF  /MOD  —>  LN-off  BLK-off
BLK#  BLK-off +  —>  Block# with line
Buff-addr.  LN-off  +  —>  LN-addr
LN-addr.  count/length  —>  ( process data)
```

Multiplying the Line# by the Line-length gives the absolute offset from the beginning of the line array to the line start. The /MOD with Bytes/block results in a Remainder/Quotient. The quotient is an offset from the starting block, BLK#. BLK# is also the start of the line array in virtual memory, and contains lines 0 to 15. The next block has lines 16 to 31, etc. The block offset is added to the BLK#. In this way lines above 15 go to the next block. The Remainder gives you the number of bytes from the start of the buffer to the line. Running BLOCK returns the buffer address, and adding the remainder to it gives the starting address of the line.

The C/L is then placed on the stack. This address and length are used by -TRAILING, which removes trailing blanks, and TYPE, which prints the message.

The B/SCR * is required in figForth because the block buffer can be smaller than the display screen, and should be removed for F83 Forth.

If you consider the messages as a fixed length Record in a random access database beginning at 'n' block, you will get a better grasp of how Forth can manage disk data.

This calculation is generic, and can be used in many situations. Once the data start address and count are obtained, the data can be retrieved directly without the need to scan from the beginning of the file. Instead of a message, the data could just as easily have been a Field in a Record, a dimension in a numeric array, or a module of binary data. Because Forth neatly separates data into blocks, data of different types can be stored in the same file without conflict.

In 'Starting Forth', Leo Brodie takes just three screens to demonstrate a simple database that could be used for a mailing list.

Conclusion

Forth blocks are basic to figForth and F83, the most common Forths for eight bit systems. You will require some familiarity to become comfortable with them. It's worth the effort because they provide the entree to the Forth programming environment which was designed for ease in testing and debugging code.

Forth is also a great experimenters language. The ability to extend Forth allows you to personalize the system, to dig around in the hardware, and to test programming algorithms.

Reference

1. Walter J. Rottenkolber: "Add and Delete Screens in PDE", Forth Dimensions, Vol.XIII, No. 1, May/June 1991, p. 23.

Source Screens

```
\ Screen 1
\ Load Include Words                    WJR01JUN95
: INCLUDE  \ ( n) (S d:filename.ext)
        FROM  LOAD ;
: INCL-THRU  \ ( from# to#) (S d:filename.ext)
        [ DOS ] FILE @ >R  OPEN  THRU  R> !FILES ;

\ Screen 3
\ Copy Routines for F83 Forth           WJR05JUN95
: 3DUP ( nnn — nnn nnn)  >R 2DUP R@ -ROT R> ;
: 3DROP  ( nnn)  DROP 2DROP ;
: RESFILE  [ DOS ] FILE @ !FILES ;
: ESTABLISH  ( n)  FILE @ SWAP  1 BUFFER# 2! ;
: (COPY)  ( from# to#)
 OFFSET @ + SWAP IN-BLOCK DROP  ESTABLISH UPDATE ;
: COPY  ( from# to#)  FLUSH (COPY)  RESFILE  FLUSH ;
  \S  Alternate Copies commented out.
: (COPY)  ( from# to#)
  SWAP IN-BLOCK SWAP BUFFER B/BUF MOVE UPDATE ;
: COPIES>  \ ( from# to# dest#) Copies from bottom up.
    -ROT 1+ SWAP DO I OVER (COPY) 1+ LOOP DROP ;
: <COPIES  \ ( from# to# dest#) Copies from top down.
    >R 2DUP SWAP - R> + -ROT  DO
    I OVER (COPY) 1-      -1 +LOOP DROP ;

\ Screen 4
\ Copy Routines for F83 Forth           WJR05JUN95
: KEY??  ( — f)  KEY? DUP IF KEY DROP THEN ;
: <COPY>  ( n n — n n)  2DUP (COPY) 1 1 ;
: COPY>  ( n n n — n n)  0 ?DO <COPY> D+ LOOP FLUSH ;
: COPIES>  \ ( from# to# dest#) Copies from bottom up.
    FLUSH -ROT 1+ OVER - #BUFFERS /MOD >R >R SWAP
    R> COPY>  R> 0            ?DO
    KEY?? ?LEAVE  #BUFFERS COPY>  LOOP 2DROP ;
: <COPY  ( n n n — n n)  0 ?DO <COPY> D- LOOP FLUSH ;
: <COPIES  \ ( from# to# dest#) Copies from top down.
   FLUSH >R 2DUP SWAP - R> + -ROT
   TUCK 1+ SWAP - #BUFFERS /MOD >R >R SWAP
   R> <COPY  R> 0            ?DO
   KEY?? ?LEAVE  #BUFFERS <COPY  LOOP 2DROP ;
\S   A Keypress Exits Copies Routine

\ Screen 5
\ Copy Routines for F83 Forth           WJR05JUN95
: (COPIES)  ( from# to# dest#)
    >R OVER R@ =  FILE @ IN-FILE @ = AND IF
    R> 3DROP EXIT                    THEN  \ Not to itself
    OVER R@ - R> SWAP 0<
    IF <COPIES  ELSE  COPIES> THEN ;
\S  Alternate (COPIES) only copies top down if
  from# < dest#. =< to#.
: (COPIES)  ( from# to# dest#)
  >R OVER R@ =  FILE @ IN-FILE @ = AND IF
  R> 3DROP EXIT                    THEN  \ Not to itself
  2DUP R@ -ROT BETWEEN R> SWAP
  IF <COPIES  ELSE  COPIES> THEN ;

\ Screen 6
\ Wipe, Delete, & Insert Screens — F83 WJR05JUN95
: COPIES  ( from# to# dest#)  (COPIES) RESFILE ;
: WIPE  ( blk#)  BLOCK B/BUF BLANK UPDATE ;
: WIPES  ( from# to#)  1+ SWAP DO I WIPE LOOP ;
: FULSCR?  \ ( blk# — f)  f= true if text in ln# 1..15
    FALSE SWAP IN-BLOCK B/BUF BOUNDS C/L +  DO
   I C@ BL <> IF NOT LEAVE THEN          LOOP ;
: ENUFSCR? \ ( to-blk# #blk — f) f= true if enuf space
    TRUE -ROT SWAP 1+ SWAP BOUNDS  DO
    I FULSCR? IF NOT LEAVE THEN    LOOP ;
: BLKOK?  ( from# to# #blks — f)  0> -ROT <= AND ;
: ?QUITBLK  ( from# to# #blks — ... ok|exit calling Word)
    3DUP BLKOK? NOT IF 3DROP R> EXIT THEN ;

\ Screen 7
\ Wipe, Delete, & Insert Screens — F83 WJR05JUN95
: (SHFT-BLK)  ( from# to# +/-#blks)
    >R OVER R> + (COPIES) ;
    \ ( alt. ver) 2 PICK + (COPIES) ;
: SHFT-BLK  ( from# to# +/-#blks) (SHFT-BLK) RESFILE ;
    \ : NIP  ( n1 n2 n3 — n1 n3)  SWAP DROP ;
: (ADDSCR)  ( start# to# #blks)
    ?QUITBLK  2DUP RESFILE  ENUFSCR?          IF
```

Continued on page 45

Special Feature

Classic Support

Apple][

High-Speed Serial I/O for the PCPI AppliCard

By John D. Baker

From the E-Mail message from John D. Baker that pretty much gets you going in the right direction on his article. BDK.

Below is the text of an article describing an SIO expansion board for the PCPI Applicard. I will follow this letter with another one which provides a schematic drawing in GIF format. If there is a problem with either of these, please let me know.

I would like to indicate that the article text and schematic image file were uploaded to my e-mail site using the SIO board described in the article.

Introduction

Classic systems tend to fall short when it comes to serial-port performance. Although unenhanced Kaypros seem to be the usual example, this problem especially plagues PCPI AppliCard-equipped Apple][systems when using CP/M-based communications software.

Recently, I designed and built a serial-port expansion board for my PCPI AppliCard systems which vastly improves serial-port performance. In the next few pages, I hope to present the results of my work and invite feedback from any other AppliCard users who construct a similar device.

Overview of the PCPI AppliCard/Apple][system

The PCPI AppliCard is a co-processor accessory card for the Apple][series of personal computers. It includes a Z80 CPU running at either 4 or 6 MHz, 64K of DRAM, bootstrap EPROM, an optional Z80 CTC, and all the necessary logic and buffers to interface the card to the Apple][expansion bus. CP/M 2.2, the Z80 CBIOS and user applications run in the RAM on the AppliCard itself.

The Apple][functions as an I/O server, providing the AppliCard access to the Apple's screen, disk drives and any other peripheral cards in the Apple for which a device driver exists. The Apple][and the AppliCard communicate through a set of I/O ports and a command processor which runs continuously on the Apple][. Unlike other systems, the AppliCard and Apple][do not share memory and there is no way to directly access the memory of one machine from the other.

This distributed system formed by the AppliCard and the Apple][allows both processors to run at full speed. As long as there is no I/O to be done, the AppliCard and Apple][are completely independent of each other. Although satisfactory for handling all in-system data transfers, this I/O bottleneck, the polled operation of the 6502 command processor, and the time required to scroll the Apple][screen severely limit serial port performance.

Normal serial I/O on the AppliCard system

The traditional arrangement has CP/M communication software read/write the command, control, status and data ports of an Apple][serial interface card indirectly, via a complex transaction with the 6502 command processor. It is common for a terminal program to lose data when the serial port is operating any faster than 2400 bits per second. File transfers, however, can operate at significantly higher speeds.

There have been a few attempts to improve the serial performance through improved device drivers which use interrupt-driven I/O and maintain a receive FIFO on the Apple][side of the system. I don't know how widely available these were or are and I haven't yet gained the skill and confidence needed to try them myself. The machine-specific overlay for such a system has to send commands to the 6502 device driver handling the serial port.

A hardware solution

Personal Computer Products, Inc. (PCPI) had planned for making expansion of the AppliCard possible. Ray Klein, the AppliCard's designer, designed and built a few serial I/O expansion devices that interfaced directly to the AppliCard, but very few of these cards were built and are thus virtually unheard-of today.

Ever since I learned of the existence of the "Klein SIO," I wanted one. However, I knew that I'd never be able to find one, so I'd have to build one of my own. With a little bit of study, scrounging for documentation and a lot of confidence- building, I finally managed to do just that.

SIO Expansion for the PCPI AppliCard

The SIO expansion device I designed and built was inspired by

the "Klein SIO." Considering that I've never seen or used one, how can I make this claim? I managed to come across the IMP and MODEM 740 overlays for the "Klein SIO" on the former SIMTEL-20 FTP site. They are still available on 'oak.oakland.edu' in the CP/M archives. The filename is 'PCPI-SIO.LBR'. Reading these overlays told me nearly everything about how the "Klein SIO" was implemented.

To begin with, my board, like the "Klein SIO," requires that a Z80 CTC be present on the AppliCard. If you were wondering what goes in that empty 28-pin DIP socket near the top of your AppliCard, it's the Z80 CTC.

I based my design on the Z80 SIO/0 for two reasons. First, I happened to have a couple of SIO/0 chips on hand. Second, I could replace them with the Z80 DART chip which has the identical pin out (except that the SIO/0's SYNCH* lines are relabeled as the DART's RI* lines) they are programmed identically for asynchronous I/O. The DART lacks synchronous I/O capability.

Any of the Z80 SIO chips (SIO/0/1/2) may be used, but the accompanying schematic will have to be altered to reflect the pinout of the particular variant used. The SIO/3/4 could also be used if you have facilities for working with QFP or PCC packages, respectively. (As it turned out, I ended up using the DART chip since I damaged both of my SIO/0's when I was building the experimental circuit on my breadboard.)

The bit-rate clock is provided by a 1.8432 MHz clock implemented as shown on the schematic. A sealed oscillator unit could be used, but I used the parts most readily available. (The "Klein SIO", in contrast, derived its bit-rate clock by dividing the system clock (6 MHz) by two.) The output of this clock drives the CLK/TRG inputs of Z80 CTC channel 0 and channel 1. The ZC/TO outputs of the CTC supply the actual bit-rate clock to the SIO transmit/receive clock inputs. By programming the CTC with appropriate time constants, bit rates from 50 to 115,200bps may be obtained.

The SIO chip is addressed as I/O at port addresses FCh-FFh. The AppliCard provides an I/O select signal (CS7* on schematic) decoded for addresses E0h-FFh. The decoding is completed by one section of a 74LS10 triple three-input NAND gate driven by address lines A2-A4. CS7* and the output of the NAND gate are combined using one section of a 74LS32 quad 2-input OR gate to provide the SIO CE* signal. Address line A1 selects which channel (A or B) of the SIO is addressed; A0 selects which register (command or data) of the SIO channel is addressed.

The only other signals that I think bear explanation are the CHAINOUT signal and the CASOUT*/CAS* signals. Quite simply, CHAINOUT is the IEO (Interrupt Enable Out) signal of the Z80 CTC installed on the AppliCard. It should, naturally, be connected to the IEI (Interrupt Enable In) signal of the SIO so the SIO can properly participate in Z80 mode-2 vectored interrupts.

CASOUT* is the output of the DRAM CAS*-generation logic. The CAS* signal, then is the CAS* input of the on-board DRAM array. In a standard 64K AppliCard, a shorting block (or jumper) is installed across pins 13 and 38 of the P2 expansion connector to complete the signal path. Any expansion device connected to P2 must maintain this connection unless the RAM expansion unit known as the "AppliDisk" is installed. As noted on the schematic, CASOUT* and CAS* should not be connected if an AppliDisk is being used.

Programming notes

The AppliCard's on-board Z80 CTC is addressed as I/O at port addresses 80h-83h. Due to partial decoding, the CTC channel control ports repeat throughout the 80h-9Fh address space. The address assignments are as follows:

Address	Function
80h	CTC channel 0 (SIO chan. A bit rate)
81h	CTC channel 1 (SIO chan. B bit rate)
82h	CTC channel 2 (unused)
83h	CTC channel 3 (Apple][Z80 mode 2 interrupt)

Programming information for the Z80 CTC can be found in the Z80 family data books.

The Z80 SIO (or DART) registers are accessed as follows:

Address	Function
FCh	SIO channel A data register
FDh	SIO channel A control register
FEh	SIO channel B data register
FFh	SIO channel B control register

Programming information for the Z80 SIO/x and DART can be found in the Z80 family data books.

The SIO expansion may operate using either polled I/O or using Z80 mode 2 interrupts. In the AppliCard system, interrupt vectors begin at FF80h. Memory addresses FF80h-FFBFh are available for interrupt service routine (ISR) addresses. The first 4 vector locations (FF80h-FF87h) are nominally reserved for the on-board CTC, although the standard system software doesn't implement CTC interrupts.

Implementation and testing notes

I built an experimental version of the SIO expansion on a breadboard and connected it to "P2" of my AppliCard with a piece of 50-line ribbon cable. A 2x25-pin IDC female header was attached to one end of the cable. On the other end, I attached 4 3M-brand 14-pin IDC DIP header connectors which I had trimmed until they fit exactly side-by-side on the 50-line ribbon cable (and in the breadboard pin holes).

Testing began by writing some code fragments using DDT's

mini assembler which initialized the CTC and SIO and read and write simple data from/to the SIO command and data registers. For feedback, I had a DVM, a simple logic probe and a modem. Seems every time I turned around, I uncovered yet another wiring error. (That's how I ruined my 2 SIO/0's—I accidentally got the SIO's D0 line plugged into +12V!)

Once satisfied that everything was working properly, I whipped up a quick overlay for QTERM to use the SIO in polled I/O mode. This was very easy to do since the addition of the SIO expansion made the AppliCard look like any other system which uses the Z80 SIO and CTC chips. I simply took the overlay for my Davidge DSB 4/6 single-board computer, changed the port addresses for the SIO and CTC, re-assembled it and patched it into QTERM. A Kaypro overlay might also be a good starting point.

Once convinced that my experimental version was working, I wire- wrapped a prototype on a 4-inch by 3-inch piece of phenolic board. The only sticky problem I had to solve was how to attach the SIO board to the "P2" expansion connector. I managed to find some SIP wire-wrap sockets and placed two 25-pin strips in adjacent rows across the phenolic board. I made sure to only wrap one level on these pins. The excess length allowed me to use 2x25 female IDC headers and a short piece of 50-line ribbon cable to connect the SIO board to the AppliCard. I used the same approach to contstruct connector "J2" shown on the schematic.

Where things stand now

During the course of extended testing, I discovered that some of my AppliCards can't cope with interrupts from the SIO board. I noticed that boards with serial numbers 13781, 800, and 580 would hang after a few characters were received. Boards with serial numbers 16582 and higher worked just fine with interrupt-driven I/O. (#16582 was the AppliCard I used during all my testing.) If any AppliCard users build a similar SIO board, I'd like to know how your system performs.

In any event, all boards worked fabulously using polled I/O. The biggest aid is that my overlays for QTERM and ZMP implement RTS/CTS handshaking and a 256-byte FIFO buffer in both polled and interrupt versions. I can run at 38400bps DTE speed using a 4MHz Applicard with polled I/O. 57600bps is easily attainable using a 6MHz AppliCard and interrupt-driven I/O. There is no character loss when using my USR Sportster 14400 modem.

I also discovered some interesting traits of the Z80 DART chip. If any data are transmitted/received prior to programming an interrupt vector (as in polled I/O), the DART must be power-cycled before it can be used in interrupt-driven I/O mode. If the RESET* line is asserted (system reset) at any time after initial power-up, the DART will not operate in interrupt-driven I/O mode. In both cases, the DART will appear to have generated an interrupt (IE0 line negated), but it never asserts the INT* line. In addition, if an interrupt vector is programmed but the DART is used in polled I/O mode, the interrupt vector cannot be re-programmed.

I couldn't find these traits mentioned in my Zilog documentation, so I don't know if they are peculiar to the DART or if the SIO also exhibits them, or if they are a side-effect of my SIO board implementation. I would appreciate feedback from anyone with more extensive experience designing with and using these parts.

The schematic indicates a real-time clock/RAM chip. That is what I had in mind to fill up the extra space the wire-wrap prototype has on the board. I have a Motorola MC146818A that I scavenged out of a dead GRiD Systems '286 laptop machine, but lately I've had my eye on a Dallas Semiconductor DS17887-5 clock/RAM module. If I had a source for the Dallas part, I'd be further inclined to complete that addition to the board.

If I had the skill and facilities, I'd like to try producing a PCB for the SIO board. That would make it much neater for installation. A big help would be to find 2x5 and 2x25 female wire-wrap header connectors that would let the board plug directly onto the AppliCard expansion connectors and pass the expansion bus on to further devices.

Closing remarks

This was the first hardware project I've designed, built, tested, and programmed entirely by myself. It wasn't until completing classes in interfacing and system design this past spring that I had the knowledge, skill, competence and, most importantly, the confidence to tackle a project like this.

Again, if any AppliCard users build or have built a similar SIO device, I'd like to hear from them about how well it worked.

John D. Baker, 3 September 1995.
Home address (semi-permanent):
Rt. 1, Box 177E
Brookshire, TX 77423
(713) 375-6522
jdb8042@blkbox.com

School address (through December 1995):
1402 Holleman, #209
College Station, TX 77840
(409) 696-0704 or -0835
jdb8042@tam2000.tamu.edu
http://tam2000.tamu.edu/~jdb8042/

References

Zilog, Inc. (1994). "Z80 microprocessor family databook". DC8321-00. Zilog, Inc. 210 East Hacienda Ave. Campbell, CA. 95008-6600. (408) 370-8000.

Personal Computer Products, Inc. (1981). APPLI-CARD (schematic).

Appendix: Time constants for standard bit rates:

For bit rates of 450bps to 115,200bps, the 1.8432 MHz clock is selected by programming the CTC for counter mode and loading the appropriate time constant as follows:

Bit Rate	CTC Time Constant (SIO in x16 mode)
450	0 (=256)
600	192
710	162 (0.16% error)
900	128
1200	96
1800	64
2400	48
3600	32
4800	24
7200	16
9600	12
14400	8
19200	6
38400	3
57600	2
115200	1

For bit rates below 450, the bit rate is derived from the system clock divided by 16 or 256 by programming the CTC for timer mode with the desired prescaler value (16 or 256) and loading the appropriate time constant, derived by the formula:

$$tc = \frac{\text{clock speed (Hz)}}{16 * ps * bps}$$

where:
tc = CTC time constant
ps = prescaler value (16 or 256)
bps = bit rate in bits per second

For example, with a 6 MHz clock, a prescaler value of 16, the time constant for 300 bps is:

$$tc = \frac{6 * 10\text{^}6}{16 * 16 * 300} = 78.125$$

which gives: tc = 78 = 4Eh (0.16% error)

It should be noted that all of the above time constant tables and calculations assume that the Z80 SIO is programmed in "x16" mode. That is, the transmit and receive clocks supplied to the SIO are sixteen times the desired bit rate. The SIO may also be programmed to "x1", "x32" or "x64" modes, meaning the transmit/receive clocks are 1, 32 or 64 times the desired bit rate, respectively. The maximum data rate of the Z80 SIO or DART is one-fifth the speed of the supplied system clock (CLK).

The CTC time constant formula can then be generalized as:

$$tc = \frac{\text{source clock (Hz)}}{scm * ps * bps}$$

where:
tc = CTC time constant
scm = SIO Clock Mode (1, 16, 32, 64)
ps = CTC prescaler value (1 for counter mode, 16 or 256 for timer mode)
bps = bit rate

Using the 1.8423 MHz clock (CTC in counter mode), with the SIO in "x32" mode, the time constant for 300 bps is calculated as:

$$tc = \frac{1.8432 * 10\text{^}6}{32 * 1 * 300} = 192 \text{ (exactly)} = C0h.$$

By choosing appropriate combinations of SIO clock mode, clock source and prescaler value, the time constant to generate reasonably accurate standard bit rates may be found by the preceeding formulas.

Appendix: Glossary.

PCPI — Personal Computer Products, Inc. Makers of the AppliCard.

Z80 — CPU and related parts used in the AppliCard and many other CP/M computers. Made by Zilog, Inc.

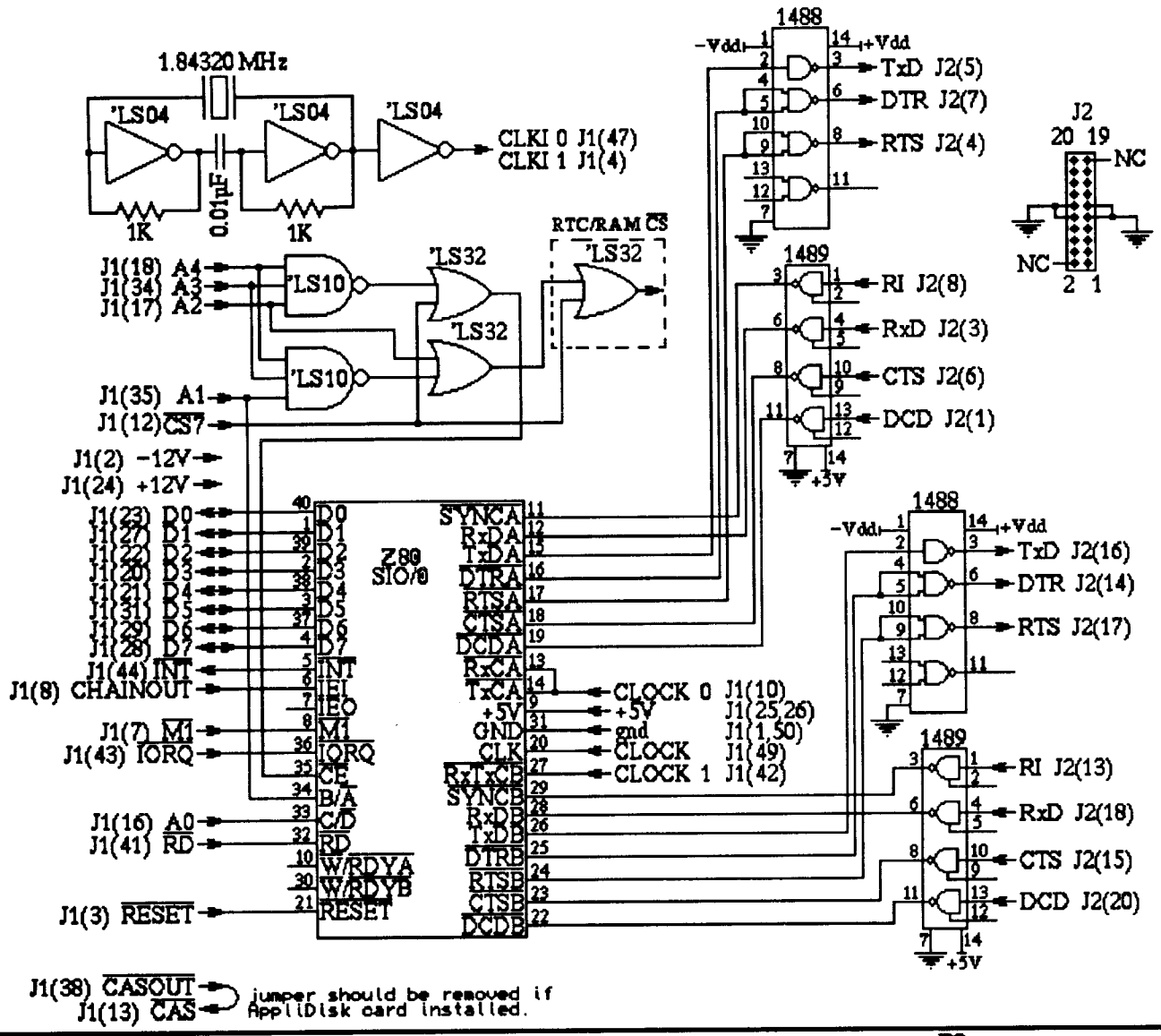
SIO — Serial Input/Output chip. Handles serial-to-parallel and parallel-to-serial data conversion.

DART — Dual Asynchronous Receiver/Transmitter. Similar to SIO, but lacks syncronous operation.

CTC — Counter/Timer Chip. Counts transitions on its clock inputs and can be programmed to interrupt or serve as a general-purpose clock divider.

bps — Bits per second. rate of data transfer. More accurate term than "baud."

PCB — Printed Circuit Board.

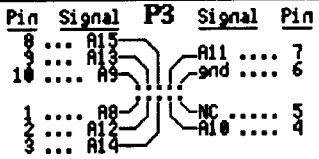


NOTES:

J1 mates with AppliCard expansion connector P2 (far right). J3 (not shown) mates with AppliCard expansion connector P3 and need only be installed if an AppliDisk card is to be piggy-backed on top of the SIO expansion board.

J2 is intended to provide 2 PC/AT-style DTE serial ports. It is constructed using 20-line ribbon cable. Install a 2x10 female IDC connector on one end. On the other end, install two IDC-style DB-9P connectors such that pin 1 aligns with line 1 (chan. A) and line 20 (chan. B) of the ribbon cable.

Decoupling capacitors are 0.1µF monolithic installed between Vcc and 0V on each IC.

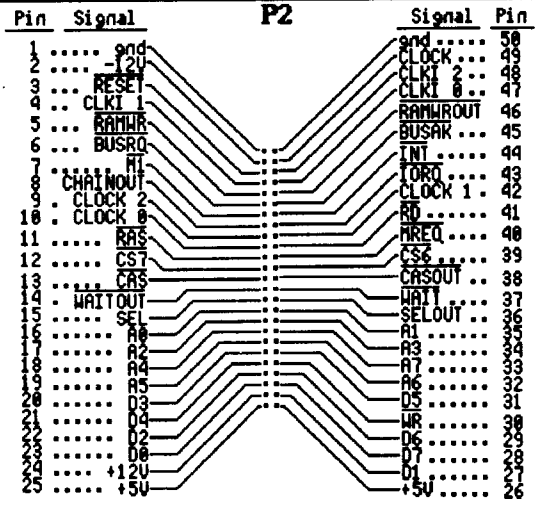


Expansion Connector Notes:

CLOCK is the system clock

CLKI n is the CLK/TRG input to the Z80 CTC channel 'n'. CLOCK n is the ZC/TO output of channel 'n' of the on-board Z80 CTC.

CAS and CASOUT are connected by a jumper or shorting block unless an AppliDisk card is installed. Some boards may have CAS and CASOUT connected by a trace on the solder side of the board.



The Z80 DART (Z8470) may be directly substituted in place of the Z80 SIO/0 (Z8440). Choose a part rated appropriately for the clock speed of your AppliCard. Interfacing to an accelerated AppliCard may require extra circuitry or higher-speed parts.

SIO Expansion for PCPI AppliCard | Designed/Drawn by: John D. Baker | Date: 28 August 1995

T9600 Source Code

By Calvin McCarthy

Special Feature

Classic Support

Last of Small Tools

The following code is from Calvin McCarthy's Tools for the F68HC11, as used by New Micros in their products. The articles appeared in issues 72 and 73. There hasn't been room to print the listing before now. You can find the code and text on a number of BBS's and on-line services. The TCJ/DIBS BBS has it as "HC11ART.ZIP".

I had to edit the code from screens and hopefully it is correct as listed. It appears that some changes were made from listings given to me, and those from which this text was made. You also might want to contact Calvin or Frank Sergant for the latest version of the 68HC11 code and PYGMY. BDK.

```
( 68HC11/F68HC11  DEVELOPMENT TOOLS                24Jul94CMc )
( DEVELOPED BY CALVIN MCCARTHY                                 )
(          12 WEDGEWOOD CRES.                                  )
(          GLOUCESTER, ONTARIO, CANADA  K1B 4B4                )
( F68HC11 DEVELOPMENT TOOLS COPYRIGHT 24 JULY, 1994            )
( TOOLS TO BE RUN ON PYGMY FORTH FOR PCs WITH                  )
( PYGMY EDITOR AND PC INTERRUPT DRIVEN SERIAL PORT             )
( SOFTWARE FROM  BRADTOOL.SCR                                  )
( TOOLS INCLUDED:                                              )
(  T9600 - TERMINAL PROGRAM WITH ECHO AND                      )
(           END-OF-LINE WAIT                                   )
(  >SBC  - F68HC11 FORTH CODE LOADER FROM PYGMY BLKS           )
(  FILE>SBC - F68HC11 LOADER FROM FILES; S19, BIN, ASCII       )

( TERMINAL FOR 68HC11                                  1Apr94CMc )
( READS ANY CHARACTER FROM F68HC11 AND DISPLAYS IT             )
( SENDS CHARACTER ENTERED AT KEYBOARD TO F68HC11               )
: T9600  9600-BAUD  START-SIO
        BEGIN       ( LOOP UNTIL esc PRESSED )
                GetRx DUP IF  EMIT
                        ELSE DROP
                        THEN
                                KEY? DUP IF DROP KEY DUP
                        $1B = IF    ( IF esc THEN LEAVE LOOP )
                        ELSE
                GetRx 0  ( ELSE SEND CHAR, STAY IN LOOP )
                THEN
        THEN
        UNTIL   STOP-SIO
;

( F68HC11 FORTH CODE LOADER                        24Jul94CMc)
: >SCR-ESCAPE  KEY? IF KEY $1B = IF QUIT THEN THEN ;
: CHAR_ECHO  >SCR-ESCAPE
        BEGIN GetRx DUP IF EMIT 1 THEN UNTIL ;
: END_LINE_ECHO
        BEGIN GetRx DUP IF DUP EMIT $007F AND $0A = THEN
UNTIL ;
: CHAR>SBC ( c — )
        DUP $0A =   ( suppress line feeds )
        IF DROP
        ELSE
                DUP $0D =   ( wait for line feed returned after cr)
                IF     PutTx  END_LINE_ECHO
                ELSE   PutTx  CHAR_ECHO
                THEN
        THEN ;
( 6811 FORTH CODE LOADER                           1Apr94CMc)
( LINE_OUT SENDS LINE TO 68HC11, ECHOS, AND DISPLAYS        )
: LINE_OUT ( ADR — ) 64 -TRAILING DUP
        IF  CUR@ 5 + AT
        0 DO
                DUP I + C@ CHAR>SBC
        LOOP
        DROP $0D CHAR>SBC
        ELSE
        DROP DROP
THEN;
( 6811 FORTH CODE LOADER                           1Apr94CMc)
: 6811_BLOCK_OUT  ( BLOCK NUMBER — )
        DUP CLS 3 5 AT ." Loading Block: " .
            4 5 AT ." Hit ESC to stop download" CR CR
        BLOCK 16 0 DO DUP I 64 * + LINE_OUT LOOP DROP ;
( OUTPUTS MULTIPLE BLOCKS TO F68HC11                        )
: >SBC ( — )  9600-BAUD START-SIO
        CLS   5 0 AT
        CR ."     FROM BLOCK #: " #INPUT
        CR ."     TO   BLOCK #: " #INPUT CR
        $0D CHAR>SBC  $0D CHAR>SBC
        1 + SWAP DO
                I 6811_BLOCK_OUT
                LOOP
        STOP-SIO  T9600 ;

( O/P-TYPE SELECT                                  17Jul94CMc)
VARIABLE FILE-TYPE
: ASC  1 FILE-TYPE ! ;    ( S19 FILES eg INTHOOK.S19 )
: 4TH  2 FILE-TYPE ! ;    ( FORTH TEXT FILES eg SREC.4TH  )
: BIN  3 FILE-TYPE ! ;    ( eg Binary files  BOOT6811.BIN )
: ASC>SBC ( c — )  PutTx CHAR_ECHO ;
: BIN>SBC ( c — )
        PutTx BEGIN GetRx DUP IF  $00FF AND U. 1 THEN UNTIL ;
: CHAR-OUT ( char — ) FILE-TYPE @ DUP 1 < OVER 3 > OR
        IF ABORT THEN
                BCASE  1 ASC>SBC  2 CHAR>SBC  3 BIN>SBC 0
EXIT ;
: BUFFER>SBC  ( #char-in-buffer — )
        PAD + PAD DO I C@ CHAR-OUT LOOP ;

( 68HC11 source code loading from text files       1Apr94CMc)
2VARIABLE FILE-LENGTH
2VARIABLE TO-DO
512 0 2CONSTANT BUFFER-LENGTH
: PARTLY-FULL-BUFFER?  ( — f )
        FILE-LENGTH 2@ FBLK @ POSITION@  D- 2DUP TO-DO 2!
        BUFFER-LENGTH  D< ;
: FULL-BUFFER  ( — # )
        PAD BUFFER-LENGTH DROP FBLK @ FILE-READ
```

```
        BUFFER-LENGTH DROP ;
: PART-FULL-BUFFER  ( — # )
        PAD TO-DO 2@ DROP FBLK @ FILE-READ
        TO-DO 2@ DROP ;

( 68HC11 source code loading from text files        1Apr94CMc)
( F>SBC fills a buffer at PAD, then sends the buffer to SBC   )
( It loops until there is less than one buffer full,          )
( sends the last buffer to the SBC, then ends the loop.       )
: F>SBC
        FBLK @ FILE-SIZE  FILE-LENGTH 2!  ( SAVE FILE-SIZE )
        FBLK @ >BOF                  ( RESET COUNT )
        BEGIN
                PARTLY-FULL-BUFFER?
                IF   PART-FULL-BUFFER  BUFFER>SBC
                        1  ( 1 FORCES END OF LOOP)
                ELSE  FULL-BUFFER  BUFFER>SBC
                        0 ( 0 REPEATS LOOP )
                THEN
        UNTIL
;

( 68HC11 FILE LOADER                          1Apr94CMc)
2VARIABLE SAVE>IN
2VARIABLE SAVE>FIN
: OPEN-FILE
   >IN 2@ SAVE>IN 2!  >FIN 2@ SAVE>FIN 2!  0 0 >FIN 2!  ( name)
        FOPEN ( handle flag) ABORT" file?" ( handle) FBLK ! ( ) ;
: CLOSE-FILE
        10 BASE !   FBLK @ FCLOSE  FIBH OFF
        SAVE>IN 2@ >IN 2! SAVE>IN 2@ >IN 2! ;

( 68HC11    FILE LOADER                       17Jul94CMc)
( ie  1200-BAUD BIN " BOOT6811.BIN" FILE>SBC )
( ie  9600-BAUD 4TH " SREC.4TH" FILE>SBC  )
( ie  9600-BAUD ASC " FFFF.S19" FILE>SBC  )
: FILE>SBC ( baud-rate file-type name — )
        OPEN-FILE
        START-SIO
        F>SBC
        STOP-SIO
        CLOSE-FILE ;
( eg 1200-BAUD  BIN SBC-INCLUDE BOOT6811.BIN )
( eg 9600-BAUD  4TH SBC-INCLUDE SREC.4TH )
( eg 9600-BAUD  ASC SBC-INCLUDE FFFF.S19 )
: SBC-INCLUDE
        32 WORD  0 OVER COUNT + C! ( a) FILE>SBC  ;

( TOOLS PROMPT FOR 68HC11 TOOLS                24Oct94CMc)
: TOOLS  CR
        ." Tools for F68HC11 Development" CR
        ." ************************************************" CR
        ." T9600 - Terminal" CR
        ." >SBC  - Download of Forth code in Pygmy blocks" CR
        ." FILE>SBC - DOWNLOAD OF DISK FILES"  CR
        ." 9600-BAUD 4TH "   $22 EMIT ." FILENAME.EXT"
                        $22 EMIT ." FILE>SBC" CR
        ." 9600-BAUD S19 "$22 EMIT ." FILENAME.EXT"
                        $22 EMIT ." FILE>SBC" CR
        ." 1200-BAUD BIN "$22 EMIT ." FILENAME.EXT"
                        $22 EMIT ." FILE>SBC" CR
        ." ************************************************"
        CR ;

( LOAD BLOCK FOR TOOLS                         27Oct94CMc)
        " PYGMY.SCR" 0 OPEN
        " PYGTOOLS.SCR" 12 OPEN
        " BRADTOOL.SCR" 13 OPEN
   13001 LOAD ( SETUP CODE NEEDED BY SERIAL PORT CODE )
        12023 LOAD  ( DO LOOP +LOOP I J K )
        12024 LOAD  ( 2CONSTANT 2VARIABLE )
        12027 LOAD  ( BCASE )
        12036 12038 THRU ( 2SWAP D+ D- D< )
        13013 LOAD
        13021 LOAD  ( STRUCTURES )
        13057 LOAD  ( SERIAL COMM ROUTINES )
        13041 LOAD  ( COMM PORT INITIALIZATION )
        168   LOAD  ( #INPUT )
        5001 5010 THRU ( F68HC11 TOOLS )

( Development LOAD                             14Sep93LGL)
        BLK @  3 + LOAD  ( +LOAD  REPEAT, )
        4  +LOAD  (  ((  OFFSET  )
        ( EXIT  ( Skip if already using PYGTOOLS )
        ( " PYGTOOLS.SCR"  12 OPEN    )
        12080 LOAD  ( ?LOAD )
        12010 LOAD  ( GET )
: 2+  2 + ;
        12071 LOAD  ( FENCE )     (   12 ?CLOSE   )
        FROM  BRADTOOL
```

HC11ART.ZIP Contents:

6811TOOL.ZIP
PYGMY14.ZIP - PYGMY DISTRIBUTION FILES
PYGTOOLS.ZIP - PYGTOOLS DISTRIBUTION FILES
BOOT6811.BIN - MACHINE CODE FOR S1S9 FILE DOWNLOAD TO 68HC11
BOOT6811.TXT - HEX LISTING OF BOOT6811.BIN
PYG14FIX.TXT - A FEW BUG FIXES FOR PYGMY 1.4
T9600.COM - EXECUTABLE 68HC11 DEVELOPMENT TOOLS
T9600.SCR - FORTH SOURCE CODE FOR THE TOOLS
TOOL.TXT - The article describing the 68HC11 development tools
68HC11.TXT - The article extolling the virtues of the F68HC11

Note: BOOT6811.TXT - HEX Listing of BOOT6811.BIN *is missing. BDK)*

Reader to Reader, continued from page 20

Well John it sounds like you had lots of fun this summer. Wish I had half as much fun as you did. Now for N help, one of our new readers O.K Hudson said he was a pied piper and N* dealer for many years. He has the original dealer service manuals and of course went to all the training on the N* products. He is willing to fend off our reader questions and help those willing to pay for the call (816-356-6309 afternon/ evenings.)*

I got one of those 2 inch IDE's and was able to buy a regular size to small size connector. Check with some of the PC dealers, most don't carry the adapters, but some will order them for you. Should make interfacing easier, but I ran into one problem, motor shut down. I think the smaller ones have some circuit for saving power and thus stop spinning if some signal is not active. Was unable to check drive as it always shutdown before it got checked. Got any ideas?

Dear Bill:

Business: I trust my call to your answer machine to reserve ad space in the Sep/Oct issue of your journal has been received

Disk I/O in Forth, continued from page 37

```
   3DUP (SHFT-BLK) NIP OVER + 1- WIPES FLUSH  ELSE
   3DROP  CR ." No Space for Insert."        THEN ;
: (DELSCR)  ( start# to# #blks)
?QUITBLK  RESFILE >R SWAP R@ + SWAP R> NEGATE
3DUP (SHFT-BLK) OVER + 1+ SWAP WIPES DROP FLUSH ;

\ Screen 8
\ Wipe, Delete, & Insert Screens — F83 WJR05JUN95
: ADDSCR  ( start# #blks)
\ Adds #blks blank screens at start# screen.
CAPACITY 1- OVER - SWAP (ADDSCR) ;
: DELSCR  ( start# #blks)
\ Deletes #blk screens at start# screen.
   CAPACITY 1- SWAP (DELSCR) ;  —>

\ Screen 9
\ Packscreen & Settle                  WJR05JUN95
: (PACKSCR)  ( from# to#)
   1+ OVER                 DO
   KEY?? ?LEAVE
   DUP FULSCR?         IF
   1+        ELSE
   I FULSCR?    IF
   DUP I BLOCK SWAP BUFFER B/BUF MOVE UPDATE
   I WIPE  1+   THEN  THEN   LOOP
   DROP  FLUSH ;
: PACKSCR  ( from# to#)
   FILE @ IN-FILE @ =         IF
   (PACKSCR)                ELSE
   2DROP ." Open File Only."  THEN ;
: SETTLE   1 CAPACITY 1- PACKSCR ;
```

and accepted. (If not please contact me.) You may note that I had tried to contact you a few times concerning my interest and left messages on your machine. I would like to have talked to you but understand completely your time constraints. I wish I myself had time to write you a more proper letter, but let me just say that it would contain praise for your efforts on the journal. You deserve that very much and I thank you.

Enclosed is a check for $160.00 for four 1/4 page ads. Also enclosed is the artwork (artwork?) for the ads. Two reduced sizes are included (You pick the appropriate one.) and a full page copy just in case. Please place the ad at your discretion where it will get maximum effectiveness.

If we sell units and make money I have other products (Pretty neat ones, I think.) that I may advertise in your journal. (6805 stuff!) Also I hope that my ads add to the financial support you need to continue *TCJ*. However, with your informative technical journal and its equally well educated readers, and this awesome Z80 computer, I feel confident that we will all meet our objectives.

This really is a neat computer. It has many of the chips characteristic of the MSX-Standard (Remember that?) or the Coleco Adam; but better. (No bias here!) The keyboard and case are a beautiful professional design (not a toy) and there are four internal connectors for custom I/O boards. With the internal switching power supply, just add software (a video monitor or I/O?) and your design is complete. Should I mention the incredibly low price.

There are two people that I would like to get an opinion from about this computer. One would be Ron Mitchell, with his hands on experience with the Adam, I think he would appreciate this design. Second would be Jay Sage, who being a CP/M wizard, might want to help me write a CP/M BIOS for this gem in exchange for many systems. But again more important I would like to get his opinion on the machine. (Hey Jay, I play table tennis in Waltham a couple times a week, lets meet and get you a system for evaluation!)

FUN: Maybe I live a pretty isolated lifestyle, and being a self-employed embedded system designer for the past 10 years, I probably do. But it really surprised me to see the Rockwell R65F11 Centerfold in the March/April issue. I thought I was the only person in the world (outside of Rockwell employees and associates - I guess that's half the world) to know about that family of microcontrollers. In the mid-eightys I designed a simple microcontroller board around the R6501Q for in house testing. It is a great chip with lots of nice features. It took much voltage abuse and held up well. (I only fried two of them in about 4 years!) The only two negatives in designing with this chip was; 1. the Q stands for the QUIP package, a strange 64-pin through hole package and 2. the chip uses power hungry NMOS technology. Not major problems, but considerations. I might add that this chip is still available from Rockwell as well as a family of other faster (10MHZ) CMOS and PLCC or DIP versions. Former Apple][people (that includes me) should love these chips for designing. Any interest out there? Call me- (508) 755-9778.(6502 lives.... embedded!!)

Let me just say that many times I wanted to write to or for *The Computer Journal* as well as other electronic publications, concerning one thing or another, but my mismanagement of time or priorities always leads me in other directions. (Usually trying to make money with zero stress induction.) Not that I have much to say, but sometimes one wants to communicate an idea. (it's a never ending battle of Ideas vs Electronics.) In any case this working-ideology has left me somewhat author less; a status I hope to change someday. Maybe with an article in *TCJ*. Oh yes I know, such noble aspirations. But as one dark haired, accordion playing, comedian Judy once said, 'it could happen .

Sincerely,
James Pellegrini

Actually James I got it in last issue. Hope the business goes well for you and do write an article. Ron Mitchell finally checked in last week and is moving to Vancouver I think, anyway without job for awhile. Jay is very busy taking care of relatives and also might be hard to catch. You might send me one next year and I'll see what it does. Then I'll write glowing reviews of it, after all it could happen...Bill.

Regular Feature

Contact Listing

SUPPORT GROUPS FOR THE CLASSICS

TCJ Staff Contacts

TCJ Editor: Bill D. Kibler, PO Box 535, Lincoln, CA 95648, (916)645-1670, GEnie: B.Kibler, CompuServe: 71563,2243, E-mail: B.Kibler@Genie.com, tcj@psyber.com.

Z-System Support: Jay Sage,1435 Centre St. Newton Centre, MA 02159-2469, (617)965-3552, BBS: (617)965-7259; E-mail: Sage@ll.mit.edu. Also sells Z-System software.

32Bit Support: Rick Rodman, BBS:(703)759-1178 (eves),BBS/FAX (703)759-1169. CompuServe 102046,1656.

Kaypro Support: Charles Stafford, 4000 Norris Ave., Sacramento, CA 95821, (916)483-0312 (eves). Also sells Kaypro upgrades, see ad inside back cover. CompuServe 73664,2470 (73664.2470@cis).

S-100 Support: Herb Johnson, CN 5256 #105, Princeton, NJ 08543, (609)771-1503. Also sells used S-100 boards and systems, see inside back cover. E-mail: hjohnson@pluto.njcc.com.

6800/6809 Support: Ronald Anderson, 3540 Sturbridge Ct., Ann Arbor, MI 48105.

Regular Contributors:
Dave Baldwin, Voice/FAX (916)722-3877, or DIBs BBS (916) 722-5799 (use "computer", "journal", pswd "subscriber" as log on), Internet dibald@netcom.com, CompuServe 70403,2444.

Brad Rodriguez,Box 77, McMaster Univ., 1280 Main St. West, Hamilton, ONT, L8S 1C0, Canada, Genie: BJ, E-mail: bj@genie.com.

Frank Sergeant, 809 W. San Antonio St., San Marcos, TX 78666, E-mail: fs07675@academia.swt.edu.

Tilmann Reh, Germany, E-mail: tilmann.reh@hrz.uni-siegen.d400.de. Has many programs for CP/M+ and is active with Z180/280 ECB bus/Modular/Embedded computers. Microcontrollers (8051).

Helmut Jungkunz, Munich, Germany, ZNODE #51, 8N1, 300-14.4, +49.89.961 45 75, or CompuServe 100024,1545.

Ron Mitchell, Apt 1107, 210 Gloucester St., Ottawa Ontario, Canada, K2P 2K4. GEnie as R.Mitchell31, or CompuServe 70323,2267.

USER GROUPS

Connecticut CP/M Users Group, contact Stephen Griswold, PO Box 74, Canton CT 06019-0074, BBS: (203)665-1100. Sponsors Z-fests.

Sacramento Microcomputer Users Group, PO Box 161513, Sacramento, CA 95816-1513, BBS: (916)372-3646. Publishes newsletter, $15.00 membership, meetings at SMUD 6201 S st., Sacramento CA.

CAPDUG: The Capital Area Public Domain Users Group, Newsletter $20, Al Siegel Associates, Inc., PO Box 34667, Betherda MD 20827. BBS (301) 292-7955.

NOVAOUG: The Northern Virginia Osborne Users Group, Newsletter $12, Robert L. Crities, 7512 Fairwood Lane, Falls Church, VA 22046. Info (703) 534-1186, BBS use CAPDUG's.

The Windsor Bulletin Board Users' Group: England, Contact Rodney Hannis, 34 Falmouth Road, Reading, RG2 8QR, or Mark Minting, 94 Undley Common, Lakenheath, Brandon, Suffolk, IP27 9BZ, Phone 0842-860469 (also sells NZCOM/Z3PLUS).

L.I.S.T.: Long Island Sinclair and Timex support group, contact Harvey Rait, 5 Peri Lane, Valley Stream, NY 11581.

ADAM-Link User's Group, Salt Lake City, Utah, BBS: (801)484-5114. Supporting Coleco ADAM machines, with Newsletter / BBS.

Adam International Media, Adam's House, Route 2, Box 2756, 1829-1 County Rd. 130, Pearland TX 77581-9503, (713)482-5040. Contact Terry R. Fowler for information.

AUGER, Emerald Coast ADAM Users Group, PO Box 4934, Fort Walton Beach FL 32549-4934, (904)244-1516. Contact Norman J. Deere, treasurer and editor for pricing and newsletter information.

MOAUG, Metro Orlando Adam Users Group, Contact James Poulin, 1146 Manatee Dr. Rockledge FL 32955, (407)631-0958.

Metro Toronto Adam Group, Box 165, 260 Adelaide St. E., Toronto, ONT M5A 1N0, Canada, (416)424-1352.

Omaha ADAM Users Club, Contact Norman R. Castro, 809 W. 33rd Ave. Bellevue NE 68005, (402)291-4405. Suppose to be oldest ADAM group.

Vancouver Island Senior ADAMphiles, ADVISA newsletter by David Cobley, 17885 Berwick Rd. Qualicum Beach, B.C., Canada V9K 1N7, (604)752-1984.

Northern Illiana ADAMS User's Group, 9389 Bay Colony Dr. #3E, Des Plaines IL 60016, (708)296-0675.

San Diego OS-9 Users Group, Contact Warren Hrach (619)221-8246, BBS: (619)224-4878.

ACCESS, PO Box 1354, Sacramento, CA 95812, Contact Bob Drews (916)423-1573. Meets first Thurdays at SMUD 59Th St. (ed. bldg.).

Forth Interest Group, PO Box 2154, Oakland CA 94621 510-89-FORTH. International support of the Forth language, local chapters.

The Pacific Northwest Heath Users Group, contact Jim Moore, PO Box 9223, Seattle, WA 98109-0223.

The SNO-KING Kaypro User Group, contact Donald Anderson, 13227 2nd Ave South, Burien, WA 98168-2637.

SeaFOG (Seattle FOG User's Group, Formerly Osborne Users Group) PO Box 12214, Seattle, WA 98102-0214.

OTHER PUBLICATIONS

The Z-Letter, supporting Z-System and CP/M users. David A.J. McGlone, Lambda Software Publishing, 149 West Hillard Lane, Eugene, OR 97404-3057, (503)688-3563. Bi-Monthly user oriented newsletter (20 pages+). Also sells CP/M Boot disks, software.

The Analytical Engine, by the Computer History Association of California, 1001 Elm Ct. El Cerrito, CA 94530-2602. A ASCII text file distributed by Internet, issue #1 was July 1993. E-mail: kcrosby@crayola.win.net.

Z-100 LifeLine, Steven W. Vagts, 2409 Riddick Rd. Elizabeth City, NC 27909, (919)338-8302. Publication for Z-100 (a S-100 machine).

The Staunch 8/89'er, Kirk L. Thompson editor, PO Box 548, West Branch IA 52358, (319)643-7136. $15/yr(US) publication for H-8/89s.

The SEBHC Journal, Leonard Geisler, 895 Starwick Dr., Ann Arbor MI 48105, (313)662-0750. Magazine of the Society of Eight-Bit Heath computerists, H-8 and H-89 support.

Sanyo PC Hackers Newsletter, Victor R. Frank editor, 12450 Skyline Blvd. Woodside, CA 94062-4541, (415)851-7031. Support for orphaned Sanyo computers and software.

the world of 68' micros, by FARNA Systems, PO Box 321, Warner Robins, GA 31099-0321. E-mail: dsrtfox@delphi.com. New magazine for support of old CoCo's and other 68xx(x) systems.

Amstrad PCW SIG, newsletter by Al Warsh, 2751 Reche Cyn Rd. #93, Colton, CA 92324. $9 for 6 bi-monthly newsletters on Amstrad CP/M machines.

Historically Brewed, A publication of the Historical Computer Society. Bimonthly at $18 a year. HCS, 2962 Park Street #1, Jacksonville, FL 32205. Editor David Greelish. Computer History and more.

IQLR (International QL Report), contact Bob Dyl, 15 Kilburn Ct. Newport, RI 02840. Subscription is $20 per year.

QL Hacker's Journal (QHJ), Timothy Swenson, 5615 Botkins Rd., Huber Heights, OH 45424, (513) 233-2178, sent mail & E-mail, swensotc@ss2.sews.wpafb.af.mil. Free to programmers of QL's.

Update Magazine, PO Box 1095, Peru, IN 46970, Subs $18 per year, supports Sinclair, Timex, and Cambridge computers.

Other Support Businesses

Hal Bower writes, sells, and supports B/PBios for Ampro, SB180, and YASBEC. $69.95. Hal Bower, 7914 Redglobe Ct., Severn MD 21144-1048, (410)551-5922.

Sydex, PO Box 5700, Eugene OR 97405, (503)683-6033. Sells several CP/M programs for use with PC Clones ('22Disk' format/copies CP/M disks using PC files system).

Elliam Associates, PO Box 2664, Atascadero CA 93423, (805)466-8440. Sells CP/M user group disks and Amstrad PCW products. See ad inside back cover.

Discus Distribution Services, Inc. sells CP/M for $150, CBASIC $600, Fortran-77 $350, Pascal/MT+ $600. 8020 San Miguel Canyon Rd., Salinas CA 93907, (408)663-6966.

Microcomputer Mail-Order Library of books, manuals, and periodicals in general and H/Zenith in particular. Borrow items for small fees. Contact Lee Hart, 4209 France Ave. North, Robbinsdale MN 55422, (612)533-3226.

Star-K Software Systems Corp. PO Box 209, Mt. Kisco, NY 10549, (914)241-0287, BBS: (914)241-3307. SK*DOS 6809/68000 operating system and software. Some educational products, call for catalog.

Peripheral Technology, 1250 E. Piedmont Rd., Marietta, GA 30067, (404)973-2156. 6809/68000 single board system. 68K ISA bus compatible system. See inside front cover.

Hazelwood Computers, RR#1, Box 36, Hwy 94@Bluffton, Rhineland, MO 65069, (314)236-4372. Some SS-50 6809 boards and new 68000 systems.

AAA Chicago Computers, Jerry Koppel, (708)681-3782. SS-50 6809 boards and systems. Very limited quanity, call for information.

MicroSolutions Computer Products, 132 W. Lincoln Hwy, DeKalb, IL 60115, (815)756-3411. Make disk copying program for CP/M systems, that runs on CP/M sytems, UNIFROM Format-translation. Also PC/Z80 CompatiCard and UniDos products.

GIMIX/OS-9, GMX, 3223 Arnold Lane, Northbrook, IL 60062, (800)559-0909, (708)559-0909, FAX (708)559-0942. Repair and support of new and old 6800/6809/68K/SS-50 systems.

n/SYSTEMS, Terry Hazen, 21460 Bear Creek Rd, Los Gatos CA 95030-9429, (408)354-7188, sells and supports the MDISK add-on RAM disk for the Ampro LB. PCB $29, assembled PCB $129, includes driver software, manual.

Corvatek, 561 N.W. Van Buren St. Corvallis OR 97330, (503)752-4833. PC style to serial keyboard adapter for Xerox, Kaypros, Franklin, Apples, $129. Other models supported.

Morgan, Thielmann & Associates services NON-PC compatible computers including CP/M as well as clones. Call Jerry Davis for more information (408) 972-1965.

Jim S. Thale Jr., 1150 Somerset Ave., Deerfield IL 60015-2944, (708)948-5731. Sells I/O board for YASBEC. Adds HD drives, 2 serial, 2 parallel ports. Partial kit $150, complete kit $210.

Trio Company of Cheektowaga, Ltd., PO Box 594, Cheektowaga NY 14225, (716)892-9630. Sells CP/M (& PC) packages: InfoStar 1.5 ($160); SuperSort 1.6 ($130), and WordStar 4.0 ($130).

Parts is Parts, Mike Zinkow, 137 Barkley Ave., Clifton NJ 07011-3244, (201)340-7333. Supports Zenith Z-100 with parts and service.

DYNACOMP, 178 Phillips Rd. Webster, NY 14580, (800)828-6772. Supplying versions of CP/M, TRS80, Apple, CoCo, Atari, PC/XT, software for older 8/16 bit systems. Call for older catalog.

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Volume Number 1:

- Issues 1 to 9
- Serial interfacing and Modem transfers
- Floppy disk formats, Print spooler.
- Adding 8087 Math Chip, Fiber optics
- S-100 HI-RES graphics.
- Controlling DC motors, Multi-user column.
- VIC-20 EPROM Programmer, CP/M 3.0.
- CP/M user functions and integration.

Volume Number 2:

- Issues 10 to 19
- Forth tutorial and Write Your Own.
- 68008 CPU for S-100.
- RPM vs CP/M, BIOS Enhancements.
- Poor Man's Distributed Processing.
- Controlling Apple Stepper Motors.
- Facsimile Pictures on a Micro.
- Memory Mapped I/O on a ZX81.

Volume Number 3:

- Issues 20 to 26
- Designing an 8035 SBC
- Using Apple Graphics from CP/M
- Soldering & Other Strange Tales
- Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K
- Extending Turbo Pascal: series
- Unsoldering: The Arcane Art
- Analog Data Acquisition & Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC
- NEW-DOS: series
- Variability in the BDS C Standard Library
- The SCSI Interface: series
- Using Turbo Pascal ISAM Files
- The Ampro Little Board Column: series
- C Column: series
- The Z Column: series
- The SCSI Interface: Introduction to SCSI
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program to Create an Index
- Selecting & Building a System
- Introduction to Assemble Code for CP/M
- Ampro 186 Column
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Volume Number 4:

- Issues 26 to 31
- Bus Systems: Selecting a System Bus
- Using the SB180 Real Time Clock
- The SCSI Interface: Software for the SCSI Adapter
- Inside Ampro Computers
- NEW-DOS: The CCP Commands (continued)
- ZSIG Corner
- Affordable C Compilers
- Concurrent Multitasking: A Review of DoubleDOS
- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
- The Art of Source Code Generation: Disassembling Z-80 Software
- Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation
- The C Column: A Graphics Primitive Package
- The Hitachi HD64180: New Life for 8-bit Systems
- ZSIG Corner: Command Line Generators and Aliases
- A Tutor Program in Forth: Writing a Forth Tutor in Forth
- Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats
- Starting Your Own BBS
- Build an A/D Converter for the Ampro Little Board
- HD64180: Setting the Wait States & RAM Refresh using PRT & DMA
- Using SCSI for Real Time Control
- Open Letter to STD Bus Manufacturers
- Patching Turbo Pascal
- Choosing a Language for Machine Control
- Better Software Filter Design
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1
- Using the Hitachi hd64180: Embedded Processor Design
- 68000: Why use a new OS and the 68000?
- Detecting the 8087 Math Chip
- Floppy Disk Track Structure
- Double Density Floppy Controller
- ZCPR3 IOP for the Ampro Little Board
- 3200 Hackers' Language
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2
- Non-Preemptive Multitasking
- Software Timers for the 68000
- Lilliput Z-Node
- Using SCSI for Generalized I/O
- Communicating with Floppy Disks: Disk Parameters & their variations
- XBIOS: A Replacement BIOS for the SB180
- K-OS ONE and the SAGE: Demystifying Operating Systems
- Remote: Designing a Remote System Program
- The ZCPR3 Corner: ARUNZ Documentation

Issue Number 32:

- 15 copies now available -

Issue Number 33:

- Data File Conversion: Writing a Filter to Convert Foreign File Formats
- Advanced CP/M: ZCPR3PLUS & How to Write Self Relocating Code
- DataBase: The First in a Series on Data Bases and Information Processing
- SCSI for the S-100 Bus: Another Example of SCSI's Versatility
- A Mouse on any Hardware: Implementing the Mouse on a Z80 System
- Systematic Elimination of MS-DOS Files: Part 2, Subdirectories & Extended DOS Services
- ZCPR3 Corner: ARUNZ Shells & Patching WordStar 4.0

Issue Number 34:

- Developing a File Encryption System.
- Database: A continuation of the data base primer series.
- A Simple Multitasking Executive: Designing an embedded controller multitasking executive.
- ZCPR3: Relocatable code, PRL files, ZCPR34, and Type 4 programs.
- New Microcontrollers Have Smarts: Chips with BASIC or Forth in ROM are easy to program.
- Advanced CP/M: Operating system extensions to BDOS and BIOS, RSXs for CP/M 2.2.
- Macintosh Data File Conversion in Turbo Pascal.

Issue Number 35:

- All This & Modula-2: A Pascal-like alternative with scope and parameter passing.
- A Short Course in Source Code Generation: Disassembling 8088 software to produce modifiable assem. source code.
- Real Computing: The NS32032.
- S-100: EPROM Burner project for S-100 hardware hackers.
- Advanced CP/M: An up-to-date DOS, plus details on file structure and formats.
- REL-Style Assembly Language for CP/M and Z-System. Part 1: Selecting your assembler, linker and debugger.

Issue Number 36:

- Information Engineering: Introduction.
- Modula-2: A list of reference books.
- Temperature Measurement & Control: Agricultural computer application.
- ZCPR3 Corner: Z-Nodes, Z-Plan, Amstrand computer, and ZFILE.
- Real Computing: NS32032 hardware for experimenter, CPUs in series, software options.
- SPRINT: A review.
- REL-Style Assembly Language for CP/M & ZSystems, part 2.
- Advanced CP/M: Environmental programming.

Issue Number 37:

- C Pointers, Arrays & Structures Made Easier: Part 1, Pointers.
- ZCPR3 Corner: Z-Nodes, patching for NZCOM, ZFILER.
- Information Engineering: Basic Concepts: fields, field definition, client worksheets.
- Shells: Using ZCPR3 named shell variables to store date variables.
- Resident Programs: A detailed look at TSRs & how they can lead to chaos.
- Advanced CP/M: Raw and cooked console I/O.
- ZSDOS: Anatomy of an Operating System: Part 1.

Issue Number 38:

- C Math: Handling Dollars and Cents With C.
- Advanced CP/M: Batch Processing and a New ZEX.
- C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.
- The Z-System Corner: Shells and ZEX, new Z-Node Central, system security under Z-Systems.
- Information Engineering: The portable Information Age.
- Computer Aided Publishing: Introduction to publishing and Desk Top Publishing.
- Shells: ZEX and hard disk backups.
- Real Computing: The National Semiconductor NS320XX.
- ZSDOS: Anatomy of an Operating System, Part 2.

Issue Number 39:

- Programming for Performance: Assembly Language techniques.
- Computer Aided Publishing: The Hewlett Packard LaserJet.
- The Z-System Corner: System enhancements with NZCOM.
- Generating LaserJet Fonts: A review of Digi-Fonts.
- Advanced CP/M: Making old programs Z-System aware.
- C Pointers, Arrays & Structures Made Easier: Part 3: Structures.
- Shells: Using ARUNZ alias with ZCAL.
- Real Computing: The National Semiconductor NS320XX.

Issue Number 40:

- Programming the LaserJet: Using the escape codes.
- Beginning Forth Column: Introduction.
- Advanced Forth Column: Variant Records and Modules.
- LINKPRL: Generating the bit maps for PRL files from a REL file.
- WordTech's dBXL: Writing your own custom designed business program.
- Advanced CP/M: ZEX 5.0=The machine and the language.
- Programming for Performance: Assembly language techniques.
- Programming Input/Output With C: Keyboard and screen functions.
- The Z-System Corner: Remote access systems and BDS C.
- Real Computing: The NS320XX

Issue Number 41:

- Forth Column: ADTs, Object Oriented Concepts.
- Improving the Ampro LB: Overcoming the 88Mb hard drive limit.
- How to add Data Structures in Forth
- Advanced CP/M: CP/M is hacker's haven, and Z-System Command Scheduler.
- The Z-System Corner: Extended Multiple Command Line, and aliases.
- Programming disk and printer functions with C.
- LINKPRL: Making RSXes easy.
- SCOPY: Copying a series of unrelated files.

Issue Number 42:

- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
- Using BYE with NZCOM.
- C and the MS-DOS Screen Character Attributes.
- Forth Column: Lists and object oriented Forth.
- The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
- 68705 Embedded Controller Application: An example of a single-chip microcontroller application.
- Advanced CP/M: PluPerfect Writer and using BDS C with REL files.

Issue Number 43:

- Standardize Your Floppy Disk Drives.
- A New History Shell for ZSystem.
- Heath's HDOS, Then and Now.
- The ZSystem Corner: Software update service, and customizing NZCOM.
- Graphics Programming With C: Graphics routines for the IBM PC, and the Turbo C graphics library.
- Lazy Evaluation: End the evaluation as soon as the result is known.
- S-100: There's still life in the old bus.
- Advanced CP/M: Passing parameters, and complex error recovery.

Issue Number 44:

- Animation with Turbo C Part 1: The Basic Tools.
- Multitasking in Forth: New Micros F68FC11 and Max Forth.
- Mysteries of PC Floppy Disks Revealed: FM, MFM, and the twisted cable.
- DosDisk: MS-DOS disk format emulator for CP/M.
- Advanced CP/M: ZMATE and using lookup and dispatch for passing parameters.
- Forth Column: Handling Strings.
- Z-System Corner: MEX and telecommunications.

Issue Number 45:

- Embedded Systems for the Tenderfoot: Getting started with the 8031.
- The Z-System Corner: Using scripts with MEX.
- The Z-System and Turbo Pascal: Patching TURBO.COM to access the Z-System.
- Embedded Applications: Designing a Z80 RS-232 communications gateway, part 1.
- Advanced CP/M: String searches and tuning Jetfind.
- Animation with Turbo C: Part 2, screen interactions.
- Real Computing: The NS32000.

Issue Number 46:

- Build a Long Distance Printer Driver.
- Using the 8031's built-in UART for serial communications.
- Foundational Modules in Modula 2.
- The Z-System Corner: Patching The Word Plus spell checker, and the ZMATE macro text editor.
- Animation with Turbo C: Text in the graphics mode.
- Z80 Communications Gateway: Prototyping, Counter/Timers, and using the Z80 CTC.

Issue Number 47:

- Controlling Stepper Motors with the 68HC11F
- Z-System Corner: ZMATE Macro Language
- Using 8031 Interrupts
- T-1: What it is & Why You Need to Know
- ZCPR3 & Modula, Too
- Tips on Using LCDs: Interfacing to the 68HC705
- Real Computing: Debugging, NS32 Multitasking & Distributed Systems
- Long Distance Printer Driver: correction
- ROBO-SOG 90

Issue Number 48:

- Fast Math Using Logarithms
- Forth and Forth Assembler
- Modula-2 and the TCAP
- Adding a Bernoulli Drive to a CP/M Computer (Building a SCSI Interface)
- Review of BDS "Z"
- PMATE/ZMATE Macros, Pt. 1
- Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX

ssue Number 49:

- Computer Network Power Protection
- Floppy Disk Alignment w/RTXEB, Pt. 1
- Motor Control with the F68HC11
- Controlling Home Heating & Lighting, Pt. 1
- Getting Started in Assembly Language
- PMATE/ZMATE Macros, Pt. 2
- Z-System Corner/ Z-Best Software

ssue Number 50:

- Offload a System CPU with the Z181
- Floppy Disk Alignment w/RTXEB, Pt. 2
- Motor Control with the F68HC11
- Modula-2 and the Command Line
- Controlling Home Heating & Lighting, Pt. 2
- Getting Started in Assembly Language Pt 2
- Local Area Networks
- Using the ZCPR3 IOP
- PMATE/ZMATE Macros, Pt. 3
- Z-System Corner, PCED/ Z-Best Software
- Real Computing, 32FX16, Caches

Issue Number 51:

- Introducing the YASBEC
- Floppy Disk Alignment w/RTXEB, Pt 3
- High Speed Modems on Eight Bit Systems
- A Z8 Talker and Host
- Local Area Networks—Ethernet
- UNIX Connectivity on the Cheap
- PC Hard Disk Partition Table
- A Short Introduction to Forth
- Stepped Inference in Embedded Control
- Real Computing, the 32CG160, Swordfish,
- PMATE/ZMATE Macros
- Z-System Corner, The Trenton Festival
- Z-Best Software, the Z3HELP System

Issue Number 52:

- YASBEC, The Hardware
- An Arbitrary Waveform Generator, Pt. 1
- B.Y.O. Assembler...in Forth
- Getting Started in Assembly Language, Pt. 3
- The NZCOM IOP
- Servos and the F68HC11
- Z-System Corner, Programming for Compatibility
- Z-Best Software
- Real Computing, X10 Revisited
- PMATE/ZMATE Macros
- Controlling Home Heating & Lighting, Pt. 3
- The CPU280, A High Performance Single-Board Computer

Issue Number 53:

- The CPU280
- Local Area Networks
- Am Arbitrary Waveform Generator
- Zed Fest '91
- Getting Started in Assembly Language
- The NZCOM IOP

Issue Number 54:

- B.Y.O. Assembler
- Local Area Networks
- Advanced CP/M
- ZCPR on a 16-Bit Intel Platform
- Real Computing
- Interrupts and the Z80
- 8 MHZ on a Ampro
- Hardware Heavenn
- What Zilog never told you about the Super8
- An Arbitary Waveform Generator
- The Development of TDOS

Issue Number 55:

- Fuzzilogy 101
- The Cyclic Redundancy Check in Forth
- The Internetwork Protocol (IP)
- Hardware Heaven
- Real Computing
- Remapping Disk Drives through Virtual BIOS
- The Bumbling Mathmatician
- YASMEM

Issue Number 56:

- TCJ - The Next Ten Years
- Input Expansion for 8031
- Connecting IDE Drives to 8-Bit Systems
- 8 Queens in Forth
- Kaypro-84 Direct File Transfers
- Analog Signal Generation

Issue Number 57:

- Home Automation with X10
- File Transfer Protocols
- MDISK at 8 MHZ.
- Shell Sort in Forth
- Introduction to Forth
- DR. S-100
- Z AT Last!

Issue Number 58:

- Multitasking Forth
- Computing Timer Values
- Affordable Development Tools
- Mr. Kaypro
- DR. S-100

Issue Number 59:

- Moving Forth
- Center Fold IMSAI MPU-A
- Developing Forth Applications
- Mr. Kaypro Review
- DR. S-100

Issue Number 60:

- Moving Forth Part II
- Center Fold IMSAI CPA
- Four for Forth
- Debugging Forth
- Support Groups for Classics
- Mr. Kaypro Review
- DR. S-100

Issue Number 61:

- Multiprocessing 6809 part I
- Center Fold XEROX 820
- Quality Control
- Real Computing
- Support Groups for Classics
- Operating Systems - CP/M
- Mr. Kaypro 5MHZ

Issue Number 62:

- SCSI EPROM Programmer
- Center Fold XEROX 820
- DR S-100
- Moving Forth part III
- Programming the 6526 CIA
- Reminiscing and Musings
- Modem Scripts

Issue Number 63:

- SCSI EPROM Programmer part II
- Center Fold XEROX 820
- DR S-100
- Multiprocessing Part II
- 6809 Operating Systems
- Reminiscing and Musings
- IDE Drives Part II

Issue Number 64:

- Small-C?
- Center Fold last XEROX 820
- DR S-100
- Moving Forth Part IV
- Small Systems
- Mr. Kaypro
- IDE Drives Part III

Issue Number 65:

- Small System Support
- Center Fold ZX80/81
- DR S-100
- Real Computing
- European Beat
- PC/XT Corner
- Little Circuits
- Levels of Forth
- Sinclair ZX81

Issue Number 66:

- Small System Support
- Center Fold: Advent Decoder
- DR S-100
- Connecting IDE Drives
- PC/XT Corner
- Little Circuits
- Multiprocessing Part III
- Z-System Corner

Issue Number 67:

- Small System Support
- Center Fold: SS-50/SS-30
- DR S-100
- Serial Kaypro Interrupts
- Little Circuits
- Moving Forth Part 5
- European Beat

Issue Number 68:

- Small System Support
- Center Fold: Pertec/Mits 4PIO
- Z-System Corner II
- PC/XT Corner
- Little Circuits
- Multiprocessing Forth Part 4
- Mr. Kaypro

Issue Number 69:

- Small System Support
- Center Fold: S-100 IDE
- Z-System Corner II
- Real Computing
- PC/XT Corner
- DR. S-100
- Moving Forth Part 6
- Mr. Kaypro

Issue Number 70:

- Small System Support
- Center Fold: Jupiter ACE
- Z-System Corner II
- PC/XT Corner: Stepper Motors
- DR. S-100
- Multiprocessing Part 5
- European Beat

Issue Number 71:

- Computing Hero of 1994
- Small System Support
- Center Fold: Hayes 80-103A
- Power Supply Basics
- PC/XT Corner: Stepper Motors
- DR. S-100
- Moving Forth Part 7
- Mr. Kaypro
- 8048 Emulator Part 1

Issue Number 72

- Beginning PLD
- Small System Support
- Center Fold: Rockwell R65F11
- Playing With Micros
- Real Computing
- Small Tools Part 1
- DR. S-100
- Moving Forth Part 7.5
- 8048 Emulator Part 2

Issue Number 73

- $10 XT
- Small System Support
- Center Fold: 640K XT
- IDE Part 6
- Real Computing
- Small Tools Part II
- DR. S-100
- Mr. Kaypro
- PC/XT Corenr
- 8048 Emulator Part 3

Issue Number 74

- Antique or Junk
- Small System Support
- Center Fold: S-100 Power Supply
- Moving Forth part 8
- Real Computing
- AMSTRAD PCW Now
- DR. S-100
- Mr. Kaypro
- Palmtech CPUZ180
- Disk I/O in Forth

Subscriptions (CA not taxable)	U.S.	Canada/Mexico (Surface)	Canada/Mexico (Air)	Europe/Other (Surface)	Europe/Other (Air)
1year (6 issues)	$24.00	$32.00	$34.00	$34.00	$44.00
2 years (12 issues)	$44.00	$60.00	$64.00	$64.00	$84.00
Back Issues (CA tax)	add these shipping costs for each issue ordered				
Bound Volumes $20.00 ea	+$3.00	+$3.50	+$6.50	+$4.00	+$17.00
#20 thru #43 are $3.00 ea.	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50
#44 and up are $4.00ea.	+$1.25	+$1.25	+$1.75	+$2.00	+$3.50

Items: _____

_____

_____ Back Issues Total _____

_____ Shipping Total _____

California state Residents add 7.25% Sales TAX _____

Subscription Total _____

Total Enclosed _____

Name: _____

Address: _____

_____

_____

_____

Credit Card # _____-_____-_____-_____ exp ___/___

Payment is accepted by check, money order, or Credit Card (M/C, VISA, CarteBlanche, Diners Club). Checks must be in US funds, drawn on a US bank. Credit Card orders can call 1(800) 424-8825.

TCJ The Computer Journal

P.O. Box 535, Lincoln, CA 95648-0535

Phone (916) 645-1670

Regular Feature

Editorial Comment

New Editor?

The Computer Corner

By Bill Kibler

Clarification of changes seems most appropriate for this corner. Or put simply, what is going on? Now of course you might say with what?

New Editor

I finially said enough and am turning things over to Dave Baldwin as your next editor. I feel I have made good changes and *TCJ* is still alive. Whether or not those changes have been the best is still open for discussion. I do feel however I was unable to keep getting new readers and our format may need some adjustment to achive that end. Saying, I just can't do it any longer pretty much sums it up.

I have enjoyed talking and dealing with all the readers and will miss the phone converstations. I will still take and answer your letters. In fact the changes will finally bring about some of the little items that bothered some of you. I had always planned on having more than one editor and now that is happening.

For me it will be a shift back to advancing my career and finding a job that better suits me and pays more. I really put my career on hold while trying to upgrade *TCJ*, but you can't do that forever. Dave hopes to hire me back sometime in the future as full time editor, but that remains to be seen.

Will I do articles, well yes, in fact I would rather have been doing them I think than be the editor. Far too many times I wanted to talk about something and was unable to complete the research to do it. With this change, I will have the time to catch up on some old projects. Take for instance: Ethernet.

DC or RF

The other day Dave and I were discussing some old projects of mine, and using Gemini TV transmitters and recivers for extending ethernet connections came up. I bought a couple units to test and they still sit in the box. See the idea is based on the fact that TV signals are basically a 4 to 6 MHZ composite signal. By that I mean a DC signal that changes from a low to high voltage at a 4 megahertz rate. For the TV transmitter to work it must pass a sigthly higher rate than the 4 and it should be flat almost to 6 MHZ before we have major loss.

Now here is where Dave and I got talking. Dave said he thought ethernet was like modems and used a carrier on which the information rode. I said no that etherenet is a DC switched signal at a 10 MHZ rate using manchester encoding. The manchester encoding guarantees that switching or changes from + to - happen often engouh that a clock pulse can be obtained from the signal. Or put another way, that no matter what the data to be sent, there will always be some transition within two clock cycles.

I did look up a National chip, and yup the physical medium is just a switching DC signal of negative variation at a 10 MHZ rate. Having refreshed my mind, I still need to take an ethernet output and drive the TV transmitter, with a diode on the other end driving a ethernet reciever. Of course I think I could also take say a RS422 driver and do the same thing, but why not just go for broke and see what happens with ethernet.

Drives

So what happened was I got a chance to check out some idea, which was long over due. Another area I will need to research and report on is using five inch drives in place of 8 inch drives. We have talked about it many times in *TCJ*, but I still get calls. What I need is a full blown article. Especially one that shows all the pin outs and options on the different size drives.

When working for Teletek, I discovered that they used a 8 to 5 adapter board. 34 pin headers where on all the S-100 boards and if you wanted 8 inch drives, you used the adapter board. Simple to make and in fact can be wire wrapped or soldered together in about an hour. I did have one around I think but not sure where it is now. It is only the Drive Selects that need jumpering, otherwise all the lines are compatible on the two drive types.

What I see the problem as - too many different options possible. You have several different types of five inch drives, and two flavors of three and a half drives. It gets so confussing that I can see why others have problems keeping it straight. So here is yet another special article I need to grind out, getting all the options in one place, with all the pin outs and mods needed to go back and forth between sizes.

Win95

Since we also talk about advanced and current systems and problems, and since I didn't want to be the only magazine to not have talked about Win95, here it is. Media hype and then some. Ok that is my opion. Should you buy, nope. Is windows NT better, seems so. Would OS2 be a good buy, maybe or maybe not, it just depends (on what I am not sure but

how about cycles of the moon). My answer is to check out the free DOS group and see what they have to offer.

There is a second group, much like Linux working on an alternative to DOS. It is sort of like the ZCPR system, a better CP/M than CP/M ever was. Well the free DOS group is like that. Trying to break the bonds of MicroSoft and give people an alternative to going to windows to get their bugs fixed. I get a rather surprised look from people when I tell them about MicroSofts bug fixes. They just do them without telling you. Try "VER /R" and see what you get. We did it about version 5 or 6 and found revisions from a to j in one office. That is about 10 bug fix releases all under one version number and without you knowing they have done anything. At least with a free DOS group I would have easily available bug lists to check my problem against.

Actually the thing that bothers me the most about the Win95 is the MSN, or MicroSoft Network, their option to the internet. A while back one of the online service providers was caught doing marketing surveys of on line users without their knowledge. My thought is this, we have no way of knowing how many hooks Win95 has in it for them to scan your hard drive and read data without your knowledge. The operating system is so big now, and difficult to know what is happening, that finding out if they are scanning your drive while connected to MSN will be almost impossible. Of course they will just say they are trying to make sure yours is a legal copy and all, but what next?

What next

For me next is issue 76 and then a long earned vacation. Dave will be removing about 30 or 40 boxes of back issues from my garage. That will make me and my wife happier. Next will be finding some missing books (I hope) and sorting through all the old systems. I plan on bringing up a Linux station full time and playing with Netware 4.1, and oh yes the Gemini units, and maybe even fixing my FAX unit (only died last year and has been waiting ever since to be fixed - bad chip in power supply).

I think one area that has dropped too far below use is my Ham equipment. In the last three years it has gotten only about 3% use, far too little. Packet radio is now 9600 baud and I understand the servers are very much improved over the ones I last used. The main problem will just be getting back to people and friends I stopped talking with. The idea had always been to just do *TCJ* full time, I see that is not an option for me. I tried, but just couldn't quite make it happen. Oh well, better luck to Dave, and thanks for keeping *TCJ* going.

1994 Computing Hero

In June I was finally able to go to the bay area and give David Jaffe his award for being the 1994 Computing Hero. I went to the monthly Forth meeting and after lunch did my five minute introduction. David took the award and did a wonderful talk on what computing was like fiffteen years ago. David sure enjoys talking about those old wonderful days and if I keep at it long enough we might actually get him to put it down in print.

What is next and for David Baldwin to do is figure out the 1995 Computing Hero. I have seen plenty of names cross my screen when researching source code that played important roles back in the mid 70s. The question is, are they still active supporting users. If you have a name that you think needs rewarding, let Dave know now so we can get started on researching their background.

I guess one requirement is they be alive and available for personal presentation. That doesn't mean they have to live in California, any of TCJ's editors will do as presentors, anywhere in the world. Hum, that makes me wonder if I can't some how get a paid vaction to somewhere exotic....

The Last Word

Well this is not the final word for good from me, just the last few this time. Rick Rodman said he has been presently surprised about my columns and their amount of information I have given out over the years (he is doing a sorted listing of articles for us). I know I surprise myself when looking back at all the work I have done over the past 13 years. It really has been fun. But, as always, you have to stop when there is no more room on the page. Till next time, keep hacking. Bill, WA6SAZ.

Editor Bill Kibler on the left, giving David Jaffe his 1994 Computing Hero award.

TCJ CLASSIFIED

CLASSIFIED RATES!
$5.00 PER LISTING!

TCJ Classified ads are on a prepaid basis only. The cost is $5.00 per ad entry. Support wanted is a free service to subscribers who need to find old or missing documentation or software. Please limit your requests to one type of system.

Commercial Advertising Rates:

Size	Once	4+
Full	$120	$90
1/2 Page	$75	$60
1/3 Page	$60	$45
1/4 Page	$50	$40
Market Place	$25	$100/yr

Send your items to:
The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

Historically Brewed. The magazine of the Historical Computer Society. Read about the people and machines which changed our world. Buy, sell and trade "antique" computers. Subscriptions $18, or try an issue for $3. HCS, 2962 Park Street #1, Jacksonville, FL 32205

Start your own technical venture! Don Lancaster's newly updated **INCREDIBLE SECRET MONEY MACHINE II** tells how. We now have autographed copies of the Guru's underground classic for $21.50. Synergetics Press, Box 809-J, Thatcher AZ, 85552.

THE CASE AGAINST PATENTS Throughly tested and proven alternatives that work in the real world. $33.50. Synergetics Press, Box 809-J, Thatcher AZ, 85552.

FOR SALE: Two Kayrpo II Computers. Each is portable with keyboard, monochrome monitor, dual 5 1/4 floppy drives and 64K memory. Also included, one padded carrying case, RS232 cable, many orginal applications (CP/M, word processor, accounting, communications, etc.) and manuals. Whole package - postpaid $160. Call (914) 331-5440, ask for Michael. (in NY)

Help Wanted: Need following items for Vector Graphics SX-3000: 1) Maintenance manual; 2) Programmers Diskette (CP/M86); 3) Vector SX MS-DOS 2.11 Users Diskette. I have all other documentation, including CP/M, MS-DOS, Programmers guide, Concurrent CP/M users guide and a CP/M Boot disk. Also trying to locate bootdisk and manuals for Vector 4. Clarence Dolan, 4553 White Horse Trail, Rockford, IL 61101-6147.

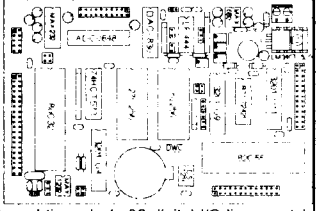
TCJ ADS WORK!

Classified ads in *TCJ*
get results, FAST!
Need to sell that special older system - TRY *TCJ*.
World Wide Coverage
with Readers interested in what **YOU** have to sell.
Provide a support service,
our readers are looking for assistance with their older systems - all the time.
The best deal in magzines,
TCJ Classified
it works!

Wanted: Have Apple IIgs, need to get hard drive and modem, but where??? David Kruszyna, 19055 Lister Ave. Eastpointe MI, 48021-2739.

VERSATILE 80C32 BASED SINGLE BOARD COMPUTER

The DC8032-1 is the first in a series of versatile single board computers based on the 8051/52 family of microcontrollers. The DC8032-1 and DC8032-2, available later this year, has been designed to provide most, if not all, of the functions required of a dedicated controller. Standard features include 32K of RAM & EPROM, A/D & D/A, real time clock, 36 digital I/O lines, watch dog timer and a centronics parallel printer port. The package also includes extended BASIC-52 with 16 additional commands, a debug monitor with 12 commands and DC TERM, a terminal emulator for communicating with the DC8032-1. A 9-volt DC wall cube is also included.



SPECIAL PRICING FOR STUDENTS

$200.00 ASSEMBLED AND ESTED, $175.00 KIT
$5.00 Shipping & Handeling + $5.00 for C.O.D.
SEND CASHIERS CHECK, MO. OR C.O.D. TO:
D. C. MICROS 1843 SUMNER CT.
LAS CRUCES, NM 88001 PH. (505) 524-4029

DIBs

Electronic Design

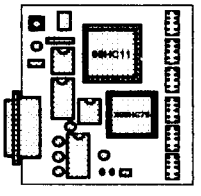
Dave Baldwin

6619 Westbrook Dr.
Citrus Heights, CA 95621

Voice/Fax (916) 722-3877
DIBs BBS (916) 722-5799

$79.95 **68HC11 Single Board Computer SBC-8K**

8K EEPROM for More Program Space!
SER-8C, Optional 8K Serial EEPROM $10



- Small Size, 3.3"x3.6"
- Low Power, <60 ma
- 8192 Bytes EEPROM
- 256 Bytes RAM
- DB-9 RS-232
- 24-TTL I/O Bits
- 8-A/D Inputs
- Power Reset Circuit
- 8 Mhz Clock
- Log Data with SER-8C

A Complete 68HC11 Development System.
New "CodeLoad+ 2.0" and Sample Programs.
No EPROMs or EPROM Programmers!
500 Pages of Manuals, 3.5" Utility Disk.

LDG Electronics
1445 Parran Road
St. Leonard, MD 20685
Voice / Fax
410-586-2177

Discover
The Z-Letter
The Z-letter is the only publication exclusively for CP/M and the Z-System. Eagle computers and Spellbinder support. Licensed CP/M distributor.

Subscriptions: $18 US, $22 Canada and Mexico, $36 Overseas. Write or call for free sample.

The Z-Letter
Lambda Software Publishing
149 West Hilliard Lane
Eugene, OR 97404-3057
(503) 688-3563

Advent Kaypro Upgrades
TurboROM. Allows flexible configuration of your entire system, read/write additional formats and more, only $35.

Replacement Floppy drives and Hard Drive Conversion Kits. Call or write for availability & pricing.

Call (916)483-0312
eves, weekends or write
Chuck Stafford
4000 Norris Ave.
Sacramento, CA 95821

TCJ MARKET PLACE
Advertising for small business
First Insertion: $25
Reinsertion: $20
Full Six Issues $100
Rates include typesetting.
Payment must accompany order.
VISA, MasterCard, Diner's Club, Carte Blanche accepted.
Checks, money orders must be US funds. Resetting of ad consitutes a new advertisement at first time insertion rates.
Mail ad or contact
The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

CP/M SOFTWARE

100 page Public Domain Catalog, $8.50 plus $1.50 shipping and handling. New Digital Research CP/M 2.2 manual, $19.95 plus $3.00 shipping and handling. Also, MS/PC-DOS Software. Disk Copying, including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog $1.00

Elliam Associates
Box 2664
Atascadero, CA 93423
805-466-8440

MORE POWER!
68HC11, 80C51 & 80C166
☑ More Microcontrollers.
☑ Faster Hardware.
☑ Faster Software.
☑ More Productive.
☑ More Tools and Utilities.
Low cost SBC's from $84. Get it done today! Not next month.
For brochure or applications:
AM Research
4600 Hidden Oaks Lane
Loomis, CA 95650
1(800) 949-8051
sofia@garlic.com

S-100/IEEE-696
IMSAI Altair
Compupro Morrow
Cromemco
and more!
Cards • Docs • Systems
Dr. S-100
Herb Johnson,
CN 5256 #105,
Princeton, NJ 08543
(609) 771-1503
hjohnson@pluto.njcc.com

THE FORTH SOURCE
Hardware & Software
MOUNTAIN VIEW PRESS
Glen B. Haydon, M.D.
Route 2 Box 429
La Honda, CA 94020
(415) 747 0760

PCB's in Minutes From LaserPrint!*
8 1/2" x 11" Sheets 100% MBG
* Or Photocopier Use household iron to apply.
Press-n-Peel
PnP BLUE or **PnP WET**
For High Precision Professional PCB Layouts
1. LaserPrint
2. Iron-On
3. Peel-Off
4. Etch
An Extra Layer of Resist for Super Fine Traces
Easy Hobby Quality PCB's
1. LaserPrint
2. Iron-On
3. Soak-Off w/ Water
4. Etch
Transfers Laser or Copier Toner as Resist
20Sh$30/40Sh$50/100Sh$100 Blue/Wet (No Mix)
Sample Pack 5 Shts Blue + 5 Shts Wet $20
VISA/MC/PO/CK/MO $4 S&H -- 2nd Day Mail
Techniks Inc. P.O. Box 463 Ringoes NJ 08551
(908)788-8249